

AD-A246 174



2

NAVAL POSTGRADUATE SCHOOL Monterey, California



DTIC
ELECTE
FEB 21 1992
S B D

THESIS

ITERATIVE METHODS FOR PARAMETER ESTIMATION

by

William R. MacHardy

December 1990

Thesis Advisor:

Murali Tummala

Approved for public release; distribution is unlimited

92-04360



92 2 19 048

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b. OFFICE SYMBOL (If applicable) EC	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School		
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000			7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State, and ZIP Code)			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
					WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) ITERATIVE METHODS FOR PARAMETER ESTIMATION					
12. PERSONAL AUTHOR(S) MACHARDY, William Robert					
13a. TYPE OF REPORT Master's Thesis		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) 1990 December	
15. PAGE COUNT 105					
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense of the US Government.					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	finite impulse response; infinite impulse response; matrix splitting; matrix partitioning; Toeplitz; symmetric, condition number		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) Starting with a least squares formulation of the parameter estimation problem, both fixed data and data-adaptive iterative algorithms are developed. We apply two new techniques, namely diagonal perturbation and multiple partitioning, to existing finite impulse response (FIR) and infinite impulse response (IIR) fixed data matrix splitting algorithms, resulting in improved performance. Also, we extend the fixed data algorithms to the data-adaptive case, and contrast them with FIR and IIR recursive least squares (RLS) algorithms. Computer simulations are used to evaluate the computational effectiveness of the new algorithms. We show the general rate of convergence for the algorithms, evaluate their ability to correctly represent the spectral components of simulated system frequency response in noise, and present system performance when the order of the model is chosen to be larger than the known system order (over-modeling).					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL TUMMALA, Murali			22b. TELEPHONE (Include Area Code) 408-646-3217		22c. OFFICE SYMBOL EC/Js

DD Form 1473, JUN 86

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

UNCLASSIFIED

Approved for public release; distribution is unlimited

Iterative Methods for Parameter
Estimation

by

William R. MacHardy
Captain, US Army
B.S, USMA, 1979

Submitted in partial fulfillment of the
requirements for the degree of

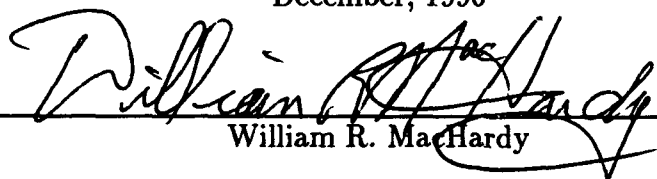
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

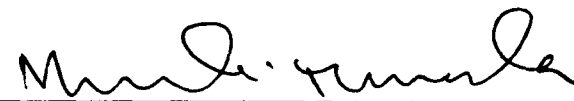
NAVAL POSTGRADUATE SCHOOL

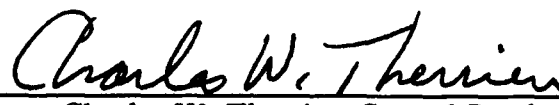
December, 1990

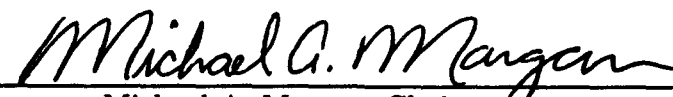
Author:


William R. MacHardy

Approved by:


Murali Tummala, Thesis Advisor


Charles W. Therrien, Second Reader


Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

Starting with a least squares formulation of the parameter estimation problem, both fixed data and data-adaptive iterative algorithms are developed. We apply two new techniques, namely diagonal perturbation and multiple partitioning, to existing *finite impulse response* (FIR) and *infinite impulse response* (IIR) fixed data matrix splitting algorithms, resulting in improved performance. Also, we extend the fixed data algorithms to the data-adaptive case, and contrast them with FIR and IIR *recursive least squares* (RLS) algorithms. Computer simulations are used to evaluate the computational effectiveness of the new algorithms. We show the general rate of convergence for the algorithms, evaluate their ability to correctly represent the spectral components of simulated system frequency response in noise, and present system performance, when the order of the model is chosen to be larger than the known system order (over-modeling).



iii

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist,	Avail and/or Special
A-1	

TABLE OF CONTENTS

I.	INTRODUCTION	1
	A. PERFORMANCE CRITERIA	1
	B. THESIS OVERVIEW	2
II.	LEAST SQUARES DATA FORMULATION	3
	A. FINITE IMPULSE RESPONSE (FIR) MODELS	4
	B. INFINITE IMPULSE RESPONSE (IIR) MODELS	6
III.	FINITE IMPULSE RESPONSE (FIR) SYSTEMS	10
	A. FIXED DATA ALGORITHMS	10
	1. Gauss-Seidel Method	10
	2. Toeplitz Approximation Algorithm	12
	3. Toeplitz Approximation with Diagonal Pertubation Algorithm	14
	4. Partitioned Toeplitz Approximation Algorithm	17
	B. DATA-ADAPTIVE ALGORITHMS	19
	1. Toeplitz Approximation Algorithm	20
	2. Recursive Least Squares (RLS) Algorithm	23
IV.	INFINITE IMPULSE RESPONSE (IIR) SYSTEMS	25
	A. FIXED DATA ALGORITHMS	25
	1. Toeplitz Approximation Algorithm	25
	2. Toeplitz Approximation with Diagonal Pertubation Algorithm	27
	3. Partitioned Toeplitz Approximation Algorithm	30
	B. DATA-ADAPTIVE ALGORITHMS	33
	1. Toeplitz Approximation Algorithm	33
	2. Recursive Least Squares (RLS) Algorithm	34

V.	CONCLUSIONS	39
A.	FIR ALGORITHMS	39
B.	IIR ALGORITHMS	40
C.	FUTURE WORK	41
APPENDIX A:	FIR SYSTEM SIMULATIONS	42
APPENDIX B:	FIXED DATA IIR SYSTEM SIMULATIONS	60
APPENDIX C:	DATA-ADAPTIVE IIR SYSTEM SIMULATIONS	70
APPENDIX D:	MATLAB CODE FOR IIR ADAPTIVE ALGORITHM	85
REFERENCES	90
INITIAL DISTRIBUTION LIST	91

LIST OF FIGURES

2.1	Fundamental System Model	3
3.1	Parameter tracks and general rate of convergence for the standard FIR Gauss-Seidel and Gauss-Seidel with SOR algorithms.	13
3.2	Parameter tracks and general rate of convergence for the FIR Toeplitz approximation and the Gauss-Seidel algorithms.	15
3.3	Parameter tracks and general rate of convergence for the FIR Toeplitz approximation and Gauss-Seidel algorithms.	16
3.4	Parameter tracks and general rate of convergence for the FIR Toeplitz approximation and Toeplitz approximation with diagonal perturbation algorithms.	18
3.5	Parameter tracks and general rate of convergence for the FIR par- titioned Toeplitz approximation and double-partitioned Toeplitz ap- proximation algorithms.	20
3.6	Parameter tracks and general rate of convergence for the FIR data- adaptive and fixed data Toeplitz approximation algorithms.	23
3.7	Parameter tracks and general rate of convergence for the FIR RLS and data-adaptive Toeplitz approximation algorithms.	24
4.1	Parameter tracks and general rate of convergence for the IIR Toeplitz approximation algorithm.	27
4.2	Parameter tracks and general rate of convergence for the IIR Toeplitz approximation with diagonal perturbation algorithm.	30
4.3	Parameter tracks and general rate of convergence for the IIR parti- tioned Toeplitz approximation algorithm.	33

4.4	Parameter tracks and general rate of convergence for the IIR data-adaptive Toeplitz approximation algorithm.	35
4.5	Frequency response of the Toeplitz approximation algorithm compared to the true system frequency response.	35
4.6	Parameter tracks and general rate of convergence for the IIR RLS algorithm.	36
4.7	Pole-zero plots for the data-adaptive IIR Toeplitz approximation algorithm with $M = N = 6$	37
4.8	Parameter track and frequency response for the data-adaptive IIR Toeplitz approximation algorithm with $M = N = 20$ and 10 dB additive noise.	38
A.1	The basic fixed data FIR Toeplitz approximation algorithm.	43
A.2	Parameter tracks and frequency response of the fixed data Toeplitz approximation with diagonal perturbation algorithm.	44
A.3	Under-modeled example of the fixed data Toeplitz approximation with diagonal perturbation algorithm.	45
A.4	Over-modeled example of the fixed data Toeplitz approximation with diagonal perturbation algorithm.	46
A.5	Over-modeled example with 10 dB of additive noise for the fixed data Toeplitz approximation with diagonal perturbation algorithm.	47
A.6	Parameter tracks and frequency response of the fixed data partitioned Toeplitz approximation algorithm.	48
A.7	Under-modeled example of the fixed data partitioned Toeplitz approximation algorithm.	49
A.8	Over-modeled example of the fixed data partitioned Toeplitz approximation algorithm.	50

A.9	Over-modeled example with 10 dB of additive noise for the fixed data partitioned Toeplitz approximation algorithm.	51
A.10	Parameter tracks and frequency response of the data-adaptive Toeplitz approximation algorithm.	52
A.11	Under-modeled example of the data-adaptive Toeplitz approximation algorithm.	53
A.12	Over-modeled example of the data-adaptive Toeplitz approximation algorithm.	54
A.13	Over-modeled example with 10 dB of additive noise for the data-adaptive Toeplitz approximation algorithm.	55
A.14	Parameter tracks and frequency response of the FIR RLS algorithm. .	56
A.15	Under-modeled example of the FIR RLS algorithm.	57
A.16	Over-modeled example of the FIR RLS algorithm.	58
A.17	Over-modeled example with 10 dB of additive noise for the data-adaptive Toeplitz approximation algorithm.	59
B.1	The basic fixed data IIR Toeplitz approximation algorithm.	61
B.2	Parameter tracks and frequency response of the fixed data Toeplitz approximation with diagonal perturbation algorithm.	62
B.3	Under-modeled example of the fixed data Toeplitz approximation with diagonal perturbation algorithm.	63
B.4	Over-modeled example of the fixed data Toeplitz approximation with diagonal perturbation algorithm.	64
B.5	Over-modeled example with 10 dB of additive noise for the fixed data Toeplitz approximation with diagonal perturbation algorithm.	65
B.6	Parameter tracks and frequency response of the fixed data partitioned Toeplitz approximation algorithm.	66

B.7	Under-modeled example of the fixed data partitioned Toeplitz approximation algorithm.	67
B.8	Over-modeled example of the fixed data partitioned Toeplitz approximation algorithm.	68
B.9	Over-modeled example with 10 dB of additive noise for the fixed data partitioned Toeplitz approximation algorithm.	69
C.1	The basic fixed data IIR Toeplitz approximation algorithm.	71
C.2	Parameter tracks and frequency response of the data-adaptive Toeplitz approximation algorithm.	72
C.3	Pole-zero plots for the data-adaptive Toeplitz approximation algorithm.	73
C.4	Under-modeled example of the data-adaptive Toeplitz approximation algorithm.	74
C.5	Pole-zero plots for the under-modeled data-adaptive Toeplitz approximation algorithm.	75
C.6	Over-modeled example of the data-adaptive Toeplitz approximation algorithm.	76
C.7	Pole-zero plots for the over-modeled data-adaptive Toeplitz approximation algorithm.	77
C.8	Over-modeled example with 10 dB of additive noise for the data-adaptive Toeplitz approximation algorithm.	78
C.9	Pole-zero plots for the over-modeled data-adaptive Toeplitz approximation algorithm with 10dB additive noise.	79
C.10	Parameter tracks and frequency response of the IIR RLS algorithm.	80
C.11	Pole-zero plots for the IIR RLS algorithm.	81
C.12	Under-modeled example of the IIR RLS algorithm.	82
C.13	Pole-zero plots for the under-modeled IIR RLS algorithm.	83

C.14 Over-modeled example of the IIR RLS algorithm.	84
---	----

ACKNOWLEDGMENT

I would like to express my gratitude and appreciation to the faculty and staff of the Electrical and Computer Engineering department for providing me with the opportunity and encouragement to explore many intriguing facets of electrical engineering. I would like to offer special thanks to Professor Murali Tummala for his guidance during my research.

I. INTRODUCTION

The general field of system modeling and parameter estimation is a rich and current area of research. In this thesis we apply four principal concepts to solving the *least squares normal equations*, which have the general form:

$$Ra = r \quad (1.1)$$

where R is the data correlation matrix, a is the parameter vector to be determined, and r is the cross-correlation vector. First, we desire to iteratively solve the problem rather than use direct inversion of the correlation matrix R . Second, we use a Toeplitz approximation matrix splitting technique [Ref. 1, 2] to set up the iterative equations. Third, we increase the diagonal dominance of the correlation matrix R in order to improve the convergence properties of the iterative equations. Finally, we partition the normal equations (1.1) in order to improve the computational efficiency and rate of convergence of the iterative algorithms. In all these cases our interest is to study both fixed data or *off-line* algorithms and data-adaptive or *online* algorithms.

A. PERFORMANCE CRITERIA

The performance criterion for the algorithms developed in this thesis is based on a least squares formulation. The algorithms minimize the sum of the squared error between the true system output and the estimated system output. We choose to use the covariance method for setting up the problem in order to remain within the available data. Although the covariance method causes the data correlation matrix R of (1.1) to be non-Toeplitz, it has the potential to provide a less biased parameter estimate than the autocorrelation method [Ref. 3].

B. THESIS OVERVIEW

This thesis is divided into five chapters, including the Introduction. In Chapter II we explicitly develop the least squares data formulation for both the *finite impulse response* (FIR) and *infinite impulse response* (IIR) system models. For the IIR system we take special care to account for the dependency of the filter output on the input and output filter coefficients. Chapter III presents iterative algorithms for FIR systems. First we consider the fixed data case, and present the simplistic Gauss-Siedel algorithm as a way of introducing the concepts of matrix splitting and iteration. Then we present the Toeplitz approximation matrix splitting algorithm, and develop two modified versions of it. Here we apply two new techniques: diagonal perturbation and multiple partitioning. Next, we develop and contrast a new data-adaptive algorithm with the well known *recursive least squares* (RLS) algorithm. IIR systems are investigated in Chapter IV. First we present the fixed data IIR Toeplitz approximation iterative algorithm, and then attempt to apply diagonal perturbation and partitioning to it. Achieving marginal success for the fixed data case, we then turn to the data-adaptive case. Here we develop a new data-adaptive IIR algorithm based on the Toeplitz approximation matrix splitting that was used in the fixed data case. This algorithm performs well, and does not have the stability problems associated with the IIR RLS algorithm, which we present for comparison. In the final chapter, we summarize the results of simulation and recommend topics for future research.

II. LEAST SQUARES DATA FORMULATION

The central problem of this thesis is to *determine the parameters of an optimal linear filter* based on the input and output measurements of a system. For the system shown in Figure 2.1, both the input data sequence $x(n)$ and the output data sequence $y(n)$ are known. The objective is to determine a linear filter, or model system which produces the output from the given input. Using least squares minimization techniques gives a filter that is optimal when it minimizes the sum of the squared error between the known output $y(n)$ and the filter output $\hat{y}(n)$. The first section of this chapter contains the least squares data formulation for the *finite impulse response* (FIR), or non-recursive filter. The next section presents the least squares data formulation for the *infinite impulse response* (IIR), or recursive filter, which has some subtle but significant differences from the FIR case. This chapter concludes with some generalizations about the algorithms to be developed.

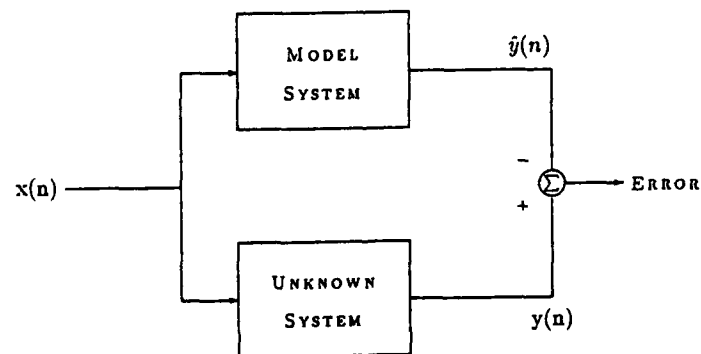


Figure 2.1: Fundamental System Model

A. FINITE IMPULSE RESPONSE (FIR) MODELS

Consider an M^{th} -order FIR filter with input $x(n)$ and output $\hat{y}(n)$. The output at time n is simply a weighted linear combination of the input:

$$\hat{y}(n) = x(n)a_0^M + x(n-1)a_1^M + \cdots + x(n-M)a_M^M. \quad (2.1)$$

This can be written as

$$\hat{y}(n) = \mathbf{x}_n \mathbf{a}^M, \quad (2.2)$$

where $\mathbf{x}_n = [x(n) \ x(n-1) \ \cdots \ x(n-M)]$ is a $1 \times M+1$ input data vector, $\mathbf{a}^M = [a_0^M \ a_1^M \ \cdots \ a_M^M]^T$ is a $M+1 \times 1$ vector containing the filter weights, or coefficients, and $\hat{y}(n)$ is the filter output at time n . The superscript M indicates that these are the coefficients of an M^{th} -order filter. Because the true value of \mathbf{a}^M is not known, equation (2.2) produces an estimate for the output, $\hat{y}(n)$. Consequently, the error $e(n)$ between the true output $y(n)$ and the FIR filter output $\hat{y}(n)$, is given by

$$e(n) = y(n) - \hat{y}(n) = y(n) - \mathbf{x}_n \mathbf{a}^M. \quad (2.3)$$

The key to a least squares solution of equation (2.2) for \mathbf{a}^M is forming an overdetermined set of $P+1$ equations (where $P \geq M$ is necessary for a unique solution):

$$\begin{aligned} \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}(n) \\ \hat{y}(n-1) \\ \vdots \\ \hat{y}(n-P) \end{bmatrix} &= \begin{bmatrix} x(n) & x(n-1) & \cdots & x(n-M) \\ x(n-1) & x(n-2) & \cdots & x(n-M-1) \\ \vdots & \vdots & & \vdots \\ x(n-P) & x(n-P-1) & \cdots & x(n-M-P) \end{bmatrix} \begin{bmatrix} a_0^M \\ a_1^M \\ \vdots \\ a_M^M \end{bmatrix} \\ \hat{\mathbf{y}} &= \begin{bmatrix} \mathbf{x}_n \\ \mathbf{x}_{n-1} \\ \vdots \\ \mathbf{x}_{n-P} \end{bmatrix} \mathbf{a}^M, \end{aligned} \quad (2.4)$$

this can be written more compactly in matrix notation as

$$\hat{\mathbf{y}} = \mathbf{X} \mathbf{a}^M. \quad (2.5)$$

Equation (2.5) is overdetermined in the sense that there are more equations than there are unknowns, or, equivalently, there are more rows of the data matrix X than there are coefficients in the filter coefficient vector \mathbf{a}^M . Using (2.5), the error *vector* can be written as: $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - X\mathbf{a}^M$. An optimal least squares solution minimizes the sum of the squared errors ζ , which is represented as:

$$\begin{aligned}\zeta &= \sum_{j=n-P}^n |e(j)|^2 = \mathbf{e}^T \mathbf{e} \\ &= (\mathbf{y} - X\mathbf{a}^M)^T (\mathbf{y} - X\mathbf{a}^M) \\ &= (\mathbf{y}^T - \mathbf{a}^{M^T} X^T) (\mathbf{y} - X\mathbf{a}^M) \\ &= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T X \mathbf{a}^M - \mathbf{a}^{M^T} X^T \mathbf{y} + \mathbf{a}^{M^T} X^T X \mathbf{a}^M.\end{aligned}\quad (2.6)$$

The minimization is accomplished by taking the derivative of the sum of the squared errors ζ with respect to the filter coefficient vector \mathbf{a}^M [Ref. 4] and setting it equal to zero:

$$\frac{\partial(\zeta)}{\partial(\mathbf{a}^M)} = 0 - 2X^T \mathbf{y} + 2X^T X \mathbf{a}^M = 0. \quad (2.7)$$

Rearranging gives

$$X^T X \mathbf{a}^M = X^T \mathbf{y} \quad (2.8)$$

as the requisite condition for ζ to be minimum. It is important to notice that the input data which forms X , and the true output data contained in \mathbf{y} are independent of the filter coefficients contained in \mathbf{a}^M . Because (2.8) contains the estimate of the optimal M^{th} -order FIR filter coefficients \mathbf{a}^M , solving (2.8) determines the optimal FIR filter. Therefore, equation (2.8) defines the FIR filter problem to be solved in Chapter III. A more standard representation of (2.8), referred to as the *normal equation* is

$$R\mathbf{a}^M = \mathbf{r}, \quad (2.9)$$

where $R = X^T X$ is referred to as the correlation matrix, and $\mathbf{r} = X^T \mathbf{y}$ is the cross-correlation of the input data matrix X and the output data vector \mathbf{y} . Theoretically

if R is full rank, the solution to (2.9) is given by $\mathbf{a}^M = R^{-1}\mathbf{r}$. This thesis presents iterative algorithms which are alternatives to the direct inversion of R . Directly inverting R to solve (2.9) has three major shortcomings:

1. Direct inversion is computationally intensive; on the order of M^3 multiplications for a $M \times M$ correlation matrix R .
2. If R is nearly singular, or very poorly conditioned it may not be possible to compute R^{-1} .
3. Unlike iterative algorithms, direct inversion cannot provide an incremental estimate of the solution.

B. INFINITE IMPULSE RESPONSE (IIR) MODELS

The general causal linear IIR filter is more complicated than the FIR filter presented above. This is because the filter output $\hat{y}(n)$ is a linear combination of the input $x(n)$, $x(n-1)$, \dots , $x(n-M)$, as well as the previous output $\hat{y}(n-1)$, $\hat{y}(n-2)$, \dots , $\hat{y}(n-N)$, where M and N are the order of the input and output coefficients, respectively. Starting with the difference equation representation of the IIR filter:

$$\begin{aligned}\hat{y}(n) &= \hat{y}(n-1)b_1^N + \hat{y}(n-2)b_2^N + \dots + \hat{y}(n-N)b_N^N \\ &+ x(n)a_0^M + x(n-1)a_1^M + \dots + x(n-M)a_M^M,\end{aligned}\quad (2.10)$$

which can be written as

$$\begin{aligned}\hat{y}(n) &= \begin{bmatrix} \hat{y}_{n-1} & \vdots & \mathbf{x}_n \end{bmatrix} \begin{bmatrix} \mathbf{b}^N \\ \vdots \\ \mathbf{a}^M \end{bmatrix} \\ \hat{y}(n) &= \mathbf{z}_n \boldsymbol{\theta},\end{aligned}\quad (2.11)$$

where $\mathbf{z}_n = [\hat{\mathbf{y}}_{n-1} : \mathbf{x}_n] = [\hat{y}(n-1) \hat{y}(n-2) \dots \hat{y}(n-N) : x(n) x(n-1) \dots x(n-M)]$ is a $1 \times N + M + 1$ data vector, $\boldsymbol{\theta} = [\mathbf{b}^{NT} : \mathbf{a}^{MT}]^T = [b_1^N b_2^N \dots b_N^N : a_0^M a_1^M \dots a_M^M]^T$

is a $N + M + 1 \times 1$ vector containing the IIR filter weights. Again, because the true value of \mathbf{b}^N and \mathbf{a}^M are not known, equation (2.11) will only produce an estimate for the output, $\hat{y}(n)$. Thus, the error between the true output $y(n)$ and the IIR filter output $\hat{y}(n)$, is given by

$$e(n) = y(n) - \hat{y}(n) = y(n) - \mathbf{z}_n \boldsymbol{\theta}. \quad (2.12)$$

As before, the least squares solution to (2.11) for $\boldsymbol{\theta}$ comes from an overdetermined set of $P + 1$ equations (where $P \geq N + M$ is necessary for a unique solution):

$$\begin{aligned} \hat{\mathbf{y}} &= \begin{bmatrix} \hat{y}(n) \\ \hat{y}(n-1) \\ \vdots \\ \hat{y}(n-P) \end{bmatrix} \\ &= \begin{bmatrix} \hat{y}(n-1) & \cdots & \hat{y}(n-N) & \vdots & x(n) & \cdots & x(n-M) \\ \hat{y}(n-2) & \cdots & \hat{y}(n-N-1) & \vdots & x(n-1) & \cdots & x(n-M-1) \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \hat{y}(n-P-1) & \cdots & \hat{y}(n-N-P) & \vdots & x(n-P) & \cdots & x(n-M-P) \end{bmatrix} \begin{bmatrix} b_1^N \\ \vdots \\ b_N^N \\ \vdots \\ a_0^M \\ \vdots \\ a_M^M \end{bmatrix} \\ \hat{\mathbf{y}} &= \begin{bmatrix} \hat{\mathbf{y}}_{n-1} & \vdots & \mathbf{x}_n \\ \hat{\mathbf{y}}_{n-2} & \vdots & \mathbf{x}_{n-1} \\ \vdots & \vdots & \vdots \\ \hat{\mathbf{y}}_{n-P-1} & \vdots & \mathbf{x}_{n-P} \end{bmatrix} \boldsymbol{\theta} = \begin{bmatrix} \mathbf{z}_n \\ \mathbf{z}_{n-1} \\ \vdots \\ \mathbf{z}_{n-P} \end{bmatrix} \boldsymbol{\theta}, \quad (2.13) \end{aligned}$$

or, in matrix notation

$$\hat{\mathbf{y}} = \mathbf{Z} \boldsymbol{\theta}, \quad (2.14)$$

where the data matrix $\mathbf{Z} = [\mathbf{z}_n^T \mathbf{z}_{n-1}^T \cdots \mathbf{z}_{n-P}^T]^T$, which is used to form an error vector, $\mathbf{e} = \mathbf{y} - \mathbf{Z} \boldsymbol{\theta}$. Once again, the optimal least squares solution minimizes the sum of the squared errors ζ , which now has the form:

$$\begin{aligned}
\zeta &= \sum_{j=n-P}^n |e(j)|^2 = \mathbf{e}^T \mathbf{e} \\
&= (\mathbf{y} - \mathbf{Z}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{Z}\boldsymbol{\theta}) \\
&= (\mathbf{y}^T - \boldsymbol{\theta}^T \mathbf{Z}^T) (\mathbf{y} - \mathbf{Z}\boldsymbol{\theta}) \\
&= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{Z}\boldsymbol{\theta} - \boldsymbol{\theta}^T \mathbf{Z}^T \mathbf{y} + \boldsymbol{\theta}^T \mathbf{Z}^T \mathbf{Z} \boldsymbol{\theta}. \quad (2.15)
\end{aligned}$$

If the gradient terms that arise from formal differentiation are neglected, the derivative of the sum of the squared errors ζ with respect to the filter coefficient vector $\boldsymbol{\theta}$ is:

$$\frac{\partial(\zeta)}{\partial(\boldsymbol{\theta})} = 0 - 2\mathbf{Z}^T \mathbf{y} + 2\mathbf{Z}^T \mathbf{Z} \boldsymbol{\theta} = 0. \quad (2.16)$$

Rearranging gives

$$\mathbf{Z}^T \mathbf{Z} \boldsymbol{\theta} = \mathbf{Z}^T \mathbf{y} \quad (2.17)$$

as the requisite condition for ζ to be minimum. Again, (2.17) contains the estimate of the IIR filter coefficients $\boldsymbol{\theta}$, and its solution determines the optimal least squares IIR filter. Therefore, equation (2.17) defines the IIR filter problem to be solved in Chapter IV, and the normal equation representation is

$$\mathbf{R}\boldsymbol{\theta} = \mathbf{r}, \quad (2.18)$$

where $\mathbf{R} = \mathbf{Z}^T \mathbf{Z}$ is referred to as the correlation matrix, and $\mathbf{r} = \mathbf{Z}^T \mathbf{y}$ is the cross-correlation vector.

The differentiation in (2.16) appears quite simple because it ignores the dependence of the filter output $\hat{y}(n)$ on the previous filter input and output. Formal differentiation of ζ gives [Ref. 5]:

$$\frac{\partial(\zeta)}{\partial(\boldsymbol{\theta})} = \frac{\partial(\zeta)}{\partial \left(\begin{bmatrix} \mathbf{b}^N \\ \mathbf{a}^M \end{bmatrix} \right)} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{b}^N} \sum_{j=n-P}^n (y(j) - \hat{y}(j))^2 \\ \frac{\partial}{\partial \mathbf{a}^M} \sum_{j=n-P}^n (y(j) - \hat{y}(j))^2 \end{bmatrix} = 0. \quad (2.19)$$

Letting $\beta_n = \frac{\partial \hat{y}(n)}{\partial \mathbf{b}^T}$ and $\alpha_n = \frac{\partial \hat{y}(n)}{\partial \mathbf{a}^T}$ represent the vector partial derivatives of the filter output, we can write

$$\sum_{j=n-P}^n \begin{bmatrix} \beta_j \\ \alpha_j \end{bmatrix} (y(j) - \mathbf{z}_j \boldsymbol{\theta}) = 0, \quad (2.20)$$

or, rearranging and letting $\psi_j = [\beta_j^T \ \alpha_j^T]^T$ gives

$$\begin{bmatrix} \sum_{j=n-P}^n \psi_j \mathbf{z}_j \end{bmatrix} \boldsymbol{\theta} = \sum_{j=n-P}^n \psi_j y(j)$$

$$\underbrace{\Psi Z}_R \boldsymbol{\theta} = \underbrace{\Psi \mathbf{y}}_r \quad (2.21)$$

so that we have the same form as the normal equations of (2.18), $R\boldsymbol{\theta} = \mathbf{r}$. However, R no longer represents the autocorrelation of the data matrix Z . In this case R represents the cross-correlation matrix between the gradient components of the filter output Ψ and the filter data matrix Z , and \mathbf{r} represents the cross-correlation between Ψ and the true output vector \mathbf{y} .

Regardless of the formulation, a standard solution to (2.18) is given by $\boldsymbol{\theta} = R^{-1}\mathbf{r}$. Again, the objective is to present iterative algorithms which are alternatives to the direct inversion of R . Finally, for the algorithms presented in this thesis, there are three general characteristics governing their suitability:

1. The algorithm should use the minimum amount of data required to produce a solution.
2. The algorithm should converge quickly. The desired number of iterations of the algorithm should not exceed 2 to 3 times the number of parameters being solved for.
3. The computational complexity of the algorithm should be less than that of direct inversion.

III. FINITE IMPULSE RESPONSE (FIR) SYSTEMS

In this chapter we consider the *finite impulse response* (FIR) system. We develop two general categories of solution algorithms, fixed data and data-adaptive. The fixed data algorithms are the Gauss-Seidel iterative method and the Toeplitz approximation iterative method. The data-adaptive algorithms are the recursive least squares (RLS) method and Toeplitz approximation method. Before developing the solution algorithms, it is worth noting that the basic problem formulation described in Chapter II is a covariance rather than a correlation formulation of the data; therefore, the matrix R in $R\mathbf{a}^M = \mathbf{r}$ is symmetric but not Toeplitz. Accordingly, we cannot apply the Levinson recursion algorithm [Ref. 6] which requires Toeplitz structure in the correlation matrix. In our formulation, the data matrix is formed using the covariance method, which has the potential to provide a less biased least squares solution than a correlation method formulation [Ref. 3].

A. FIXED DATA ALGORITHMS

1. Gauss-Seidel Method

A very simple and straightforward iterative algorithm is the Gauss-Seidel method [Ref. 7]. We drop the superscript M from \mathbf{a}^M for simplicity. Unless otherwise stated, we are considering an M^{th} -order FIR system. Starting with (2.9), $R\mathbf{a} = \mathbf{r}$, split R into $L + D + U$, where L is a matrix containing the strictly lower triangular elements of R , U is a matrix containing the strictly upper triangular elements of R , and D is a diagonal matrix containing the main diagonal elements of R . Substituting

this into (2.9) gives

$$(L + D + U)\mathbf{a} = \mathbf{r}, \quad (3.1)$$

or

$$(L + D)\mathbf{a} = -U\mathbf{a} + \mathbf{r}. \quad (3.2)$$

Making (3.2) into an iterative algorithm requires that we isolate the parameter vector \mathbf{a} on the left side of the equation. Pre-multiplying both sides of the equation by $(L + D)^{-1}$:

$$(L + D)^{-1}(L + D)\mathbf{a} = (L + D)^{-1}(-U\mathbf{a} + \mathbf{r}) \quad (3.3)$$

we have

$$\mathbf{a} = -(L + D)^{-1}U\mathbf{a} + (L + D)^{-1}\mathbf{r}. \quad (3.4)$$

Note that $(L + D)$ must be nonsingular for this technique to work. Now there are two occurrences of the parameter vector \mathbf{a} . We define the one on the right side of (3.4) as the current value $\mathbf{a}^{(k)}$, and the one on the left side as the updated value $\mathbf{a}^{(k+1)}$, where k is the iteration index. Thus, the final form of the Gauss-Seidel iterative algorithm is:

$$\mathbf{a}^{(k+1)} = -(L + D)^{-1}U\mathbf{a}^{(k)} + (L + D)^{-1}\mathbf{r}. \quad (3.5)$$

Unfortunately, this algorithm takes an excessive number of iterations to converge to the true parameter values when R is even moderately ill-conditioned. In an effort to improve this performance, we applied the successive overrelaxation (SOR) acceleration technique [Ref. 9] to (3.5). The SOR technique requires that we add an acceleration parameter ω as follows:

$$\mathbf{a}^{(k+1)} = -(\omega L + D)^{-1}[(1 - \omega)D - \omega U]\mathbf{a}^{(k)} + (\omega L + D)^{-1}\omega\mathbf{r}. \quad (3.6)$$

Notice that when ω equals 1, (3.6) degenerates into the standard Gauss-Seidel algorithm. In order to speed up the rate of convergence of the algorithm, the acceleration

parameter ω must assume its optimum value, which is given by [Ref. 9]:

$$\omega_{opt} = \frac{2(1 - \sqrt{1 - \mu_{max}^2})}{\mu_{max}^2}. \quad (3.7)$$

In (3.7) the term μ_{max} represents the largest eigenvalue of the Jacobi matrix, which has the form $D^{-1}(-L - U)$. Unfortunately, though this is a popular and usually successful acceleration technique, for the basic problem of this thesis it was found to produce little improvement in the performance of the Gauss-Seidel algorithm.

In order to observe the performance of the algorithms developed in this chapter, they are simulated with the following FIR system:

$$\begin{aligned} y(n) = & 0.5x(n-1) + 0.25x(n-2) - 0.5x(n-4) + 0.053x(n-5) + 0.0345x(n-6) \\ & - 0.76x(n-7) + 3.5x(n-8) - 1.0032x(n-9) - 0.0031x(n-10). \end{aligned} \quad (3.8)$$

Figure 3.1 shows the generally slow and gradual rate of convergence of the Gauss-Seidel and Gauss-Seidel with SOR algorithms. Each parameter track line of Figure 3.1 corresponds to a coefficient in (3.8). Notice that as the number of iterations k increases, the parameter track lines flatten out and approach the values in (3.8).

2. Toeplitz Approximation Algorithm

One reason for the slow rate of convergence of the Gauss-Seidel method is that it fails to capitalize on the structure of the covariance matrix R . The covariance matrix is symmetric but not Toeplitz. However, it does have an underlying Toeplitz structure. This point becomes clear when the number of data points used to form the matrix is increased; the covariance matrix then approaches a Toeplitz structure. Toeplitz matrices have many nice properties, and efficient methods such as the Levinson recursion are available to invert a Toeplitz matrix. An iterative algorithm which takes advantage of the near Toeplitz structure of the covariance matrix [Ref. 1] is developed by splitting R into a Toeplitz matrix T , and a residual matrix S . The matrix

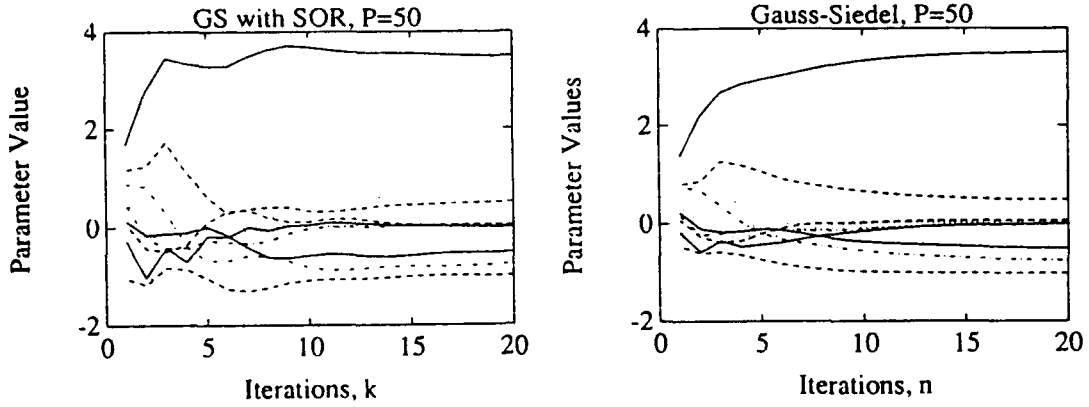


Figure 3.1: Parameter tracks and general rate of convergence for the standard FIR Gauss-Seidel and Gauss-Seidel with SOR algorithms.

T is obtained by averaging the diagonal elements of R . This Toeplitz approximation provides a natural splitting of the covariance matrix, $R = T + S$, which is used to develop an iterative algorithm as follows. Beginning with

$$Ra = r, \quad (3.9)$$

substituting for $R = T + S$ gives

$$(T + S)a = r, \quad (3.10)$$

or, rearranging we have,

$$Ta = r - Sa. \quad (3.11)$$

But $S = R - T$, thus

$$Ta = r - (R - T)a \quad (3.12)$$

and isolating a gives

$$a = T^{-1}r - T^{-1}Ra + a. \quad (3.13)$$

Finally, the iterative algorithm has the form:

$$\mathbf{a}^{(k+1)} = T^{-1}\mathbf{r} - T^{-1}R\mathbf{a}^{(k)} + \mathbf{a}^{(k)}. \quad (3.14)$$

And if we let $\mathbf{a}_0 = T^{-1}\mathbf{r}$ be the initial estimate of the parameter vector \mathbf{a} , then we have:

$$\mathbf{a}^{(k+1)} = \mathbf{a}_0 + \mathbf{a}^{(k)} - T^{-1}R\mathbf{a}^{(k)}. \quad (3.15)$$

The Toeplitz approximation matrix T can also be viewed in another light [Ref. 8]. It can be thought of as a pre-conditioning matrix for R . Indeed, part of the difficulty with the Gauss-Seidel method of the previous section is that as R becomes more ill-conditioned, the iterative algorithm takes much longer to converge. We observed that the product $T^{-1}R$ does in fact have a better condition number than R by at least an order of magnitude when R is very ill-conditioned. Thus, we might expect that the algorithm of (3.15) will converge to the true parameter vector \mathbf{a} faster than (3.5) or (3.6). This is in fact the case. In Figure 3.2 it is clear that the Toeplitz approximation algorithm gives much faster convergence to the true parameter values than the Gauss-Seidel algorithm.

3. Toeplitz Approximation with Diagonal Perturbation Algorithm

One significant shortcoming of (3.15) is that it fails to converge as the covariance matrix R becomes extremely ill-conditioned. Despite the fact that T functions as a pre-conditioner for R , the algorithm of (3.15) will not converge as the spectral radius (or largest eigenvalue) of $(T^{-1}S)$ becomes greater than one [Ref. 9]. Generally, the condition number of R is an indication of how poorly conditioned it is. We observe that as the number of data points used to form R is reduced from a large number (i.e., 5 to 10 times the order of the filter being solved for) to near the minimum number required for a unique solution (i.e., the order of the filter), the condition number of R grows quite large. Under these circumstances the algorithm of

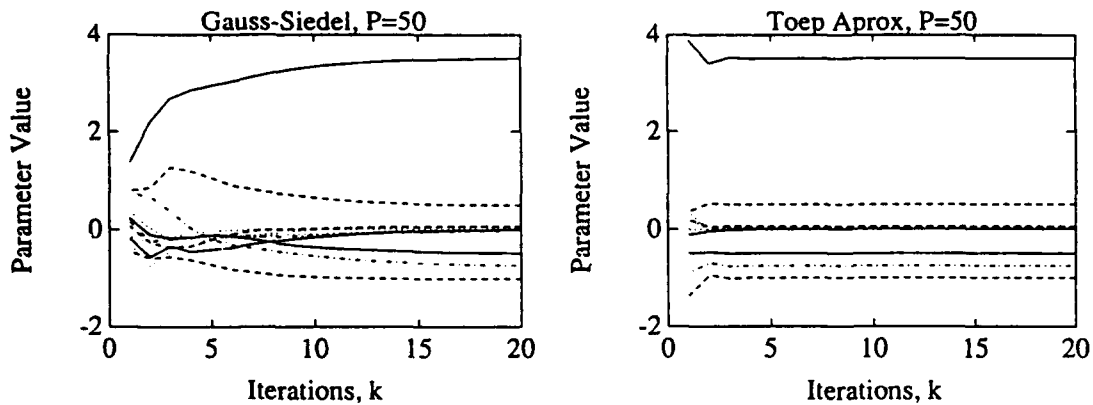


Figure 3.2: Parameter tracks and general rate of convergence for the FIR Toeplitz approximation and the Gauss-Seidel algorithms.

(3.15) does not converge. By contrast the Gauss-Seidel algorithm always converges to the true parameter values, although it takes an excessively large number of iterations for the parameter values to approach their true values (see Figure 3.3). Viewing the ill-conditioned nature of R as an obstacle to satisfactory performance of (3.15), we now introduce a technique which improves the condition of R and permits a modified version of (3.15) to converge as the number of data points approaches the minimum required.

Beginning with the fact that any identity matrix I is invertible with a condition number of one, we seek to alter R such that it becomes more like an identity matrix. This is accomplished by adding a diagonal matrix to R such that the resulting matrix has its main diagonal elements at least an order of magnitude larger than any other element in the matrix, then this new matrix has increased diagonal dominance and begins to look more like an identity matrix. The consequence of applying this technique is that we improve the condition of R and the performance of (3.15).

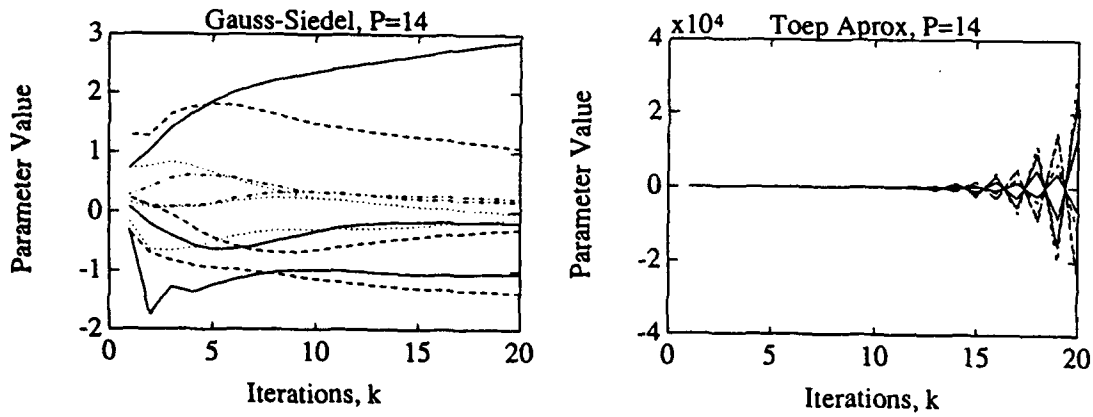


Figure 3.3: Parameter tracks and general rate of convergence for the FIR Toeplitz approximation and Gauss-Seidel algorithms.

If we let σ be a scalar multiplier and the Toeplitz approximation splitting of R be defined as $R = T_o + S$, then we can derive a modified version of (3.15) as follows. Beginning with

$$Ra = r, \quad (3.16)$$

adding a diagonal matrix (scaled by σ) to R gives

$$(R + \sigma I)a = r + \sigma a \quad (3.17)$$

and substituting for $R = T_o + S$ we have

$$(T_o + S + \sigma I)a = r + \sigma a. \quad (3.18)$$

Now, letting $T = T_o + \sigma I$, we can write

$$Ta = r + \sigma a - Sa \quad (3.19)$$

but $S = R + \sigma I - T$, and substituting gives

$$Ta = r + \sigma a - (R + \sigma I - T)a. \quad (3.20)$$

After rearranging it is easy to see that the σa terms cancel

$$Ta = r + \sigma a - \sigma a - (R - T)a \quad (3.21)$$

so that we can now write

$$a = T^{-1}r - T^{-1}Ra + a. \quad (3.22)$$

Now, letting $a_0 = T^{-1}r$ and rearranging, the final form of the Toeplitz approximation with diagonal perturbation iterative algorithm is:

$$a^{(k+1)} = a_0 + a^{(k)} - T^{-1}Ra^{(k)}. \quad (3.23)$$

Interestingly, the algorithm of (3.23) has exactly the same form as (3.15). The only difference is that the matrix T in (3.23) is the original Toeplitz approximation matrix T_0 plus some diagonal matrix σI . The new algorithm of (3.23) was observed to converge where the algorithm of (3.15) fails. It always reaches the vicinity of the true parameter values quickly, but requires a large number of iterations to converge to the exact solution (see Figure 3.4). In simulation, we observe that optimum values for σ are inversely proportional to the condition number of R . One difficulty with this algorithm is that determining the optimum value for σ is a problem as it is data dependent. Therefore, achieving optimal performance of this algorithm is dependent on finding a suitable value for σ . We were unable to find a general expression for the optimum value of σ in this thesis.

4. Partitioned Toeplitz Approximation Algorithm

An alternative to improving the condition of R in (3.15) is to block partition R . It may be partitioned into equally or unequally sized parts, however the number of row partitions must equal the number of column partitions. We now derive a partitioned version of (3.15). Beginning with

$$Ra = r, \quad (3.24)$$

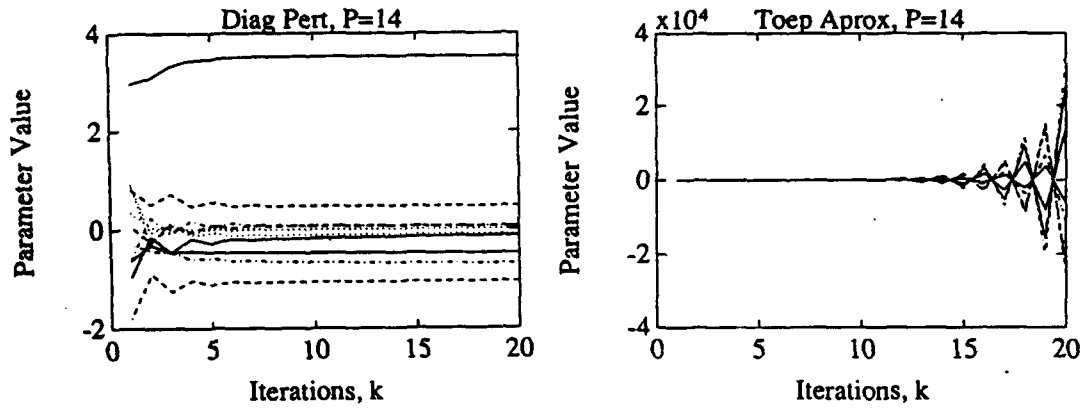


Figure 3.4: Parameter tracks and general rate of convergence for the FIR Toeplitz approximation and Toeplitz approximation with diagonal perturbation algorithms.

partitioning R into four parts gives

$$\begin{bmatrix} R_{11} & \vdots & R_{12} \\ \dots & \dots & \dots \\ R_{21} & \vdots & R_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ \vdots \\ r_2 \end{bmatrix}, \quad (3.25)$$

and substituting for $R_{11} = T_{11} + S_{11}$ and $R_{22} = T_{22} + S_{22}$ we have

$$\begin{bmatrix} T_{11} + S_{11} & \vdots & R_{12} \\ \dots & \dots & \dots \\ R_{21} & \vdots & T_{22} + S_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ \vdots \\ r_2 \end{bmatrix}. \quad (3.26)$$

Multiplying gives the following coupled equations:

$$\begin{aligned} (T_{11} + S_{11})a_1 + R_{12}a_2 &= r_1 \\ (T_{22} + S_{22})a_2 + R_{21}a_1 &= r_2 \end{aligned} \quad (3.27)$$

and rearranging this yields

$$\begin{aligned} T_{11}a_1 &= r_1 - R_{12}a_2 - S_{11}a_1 \\ T_{22}a_2 &= r_2 - R_{21}a_1 - S_{22}a_2. \end{aligned} \quad (3.28)$$

But $S_{11} = R_{11} - T_{11}$ and $S_{22} = R_{22} - T_{22}$, thus we have

$$\begin{aligned} \mathbf{a}_1 &= T_{11}^{-1} \mathbf{r}_1 - T_{11}^{-1} (R_{12} \mathbf{a}_2 + (R_{11} - T_{11}) \mathbf{a}_1) \\ \mathbf{a}_2 &= T_{22}^{-1} \mathbf{r}_2 - T_{22}^{-1} (R_{21} \mathbf{a}_1 + (R_{22} - T_{22}) \mathbf{a}_2) \end{aligned} \quad (3.29)$$

and as before we let $\mathbf{a}_{10} = T_{11}^{-1} \mathbf{r}_1$ and $\mathbf{a}_{20} = T_{22}^{-1} \mathbf{r}_2$ so that now we can write

$$\begin{aligned} \mathbf{a}_1 &= \mathbf{a}_{10} + \mathbf{a}_1 - T_{11}^{-1} (R_{12} \mathbf{a}_2 + R_{11} \mathbf{a}_1) \\ \mathbf{a}_2 &= \mathbf{a}_{20} + \mathbf{a}_2 - T_{22}^{-1} (R_{21} \mathbf{a}_1 + R_{22} \mathbf{a}_2). \end{aligned} \quad (3.30)$$

The final form of the partitioned Toeplitz approximation iterative algorithm is

$$\begin{aligned} \mathbf{a}_1^{(k+1)} &= \mathbf{a}_{10} + \mathbf{a}_1^{(k)} - T_{11}^{-1} (R_{12} \mathbf{a}_2^{(k)} + R_{11} \mathbf{a}_1^{(k)}) \\ \mathbf{a}_2^{(k+1)} &= \mathbf{a}_{20} + \mathbf{a}_2^{(k)} - T_{22}^{-1} (R_{21} \mathbf{a}_1^{(k+1)} + R_{22} \mathbf{a}_2^{(k)}). \end{aligned} \quad (3.31)$$

This partitioning algorithm was also observed to converge to the true solution where the algorithm of (3.15) fails. Additionally, this algorithm converges to the true solution in about the same number of iterations as that of (3.23), and this algorithm does not require the choice of an optimal parameter value such as σ in (3.23). It is possible to derive iterative algorithms using this same methodology beyond the two partitions shown in (3.25), however we did not observe a significant difference in performance of these algorithms over the one shown in (3.31) (see Figure 3.5). Finally, it is also possible to combine diagonal perturbation and partitioning in a single algorithm, but again we did not observe any significant improvement in performance.

B. DATA-ADAPTIVE ALGORITHMS

Although the fixed data algorithms above do provide a solution to the least squares problem $R\mathbf{a} = \mathbf{r}$, they require that a minimum amount of data be accumulated before the iterative algorithm may begin and a solution determined. Consequently,

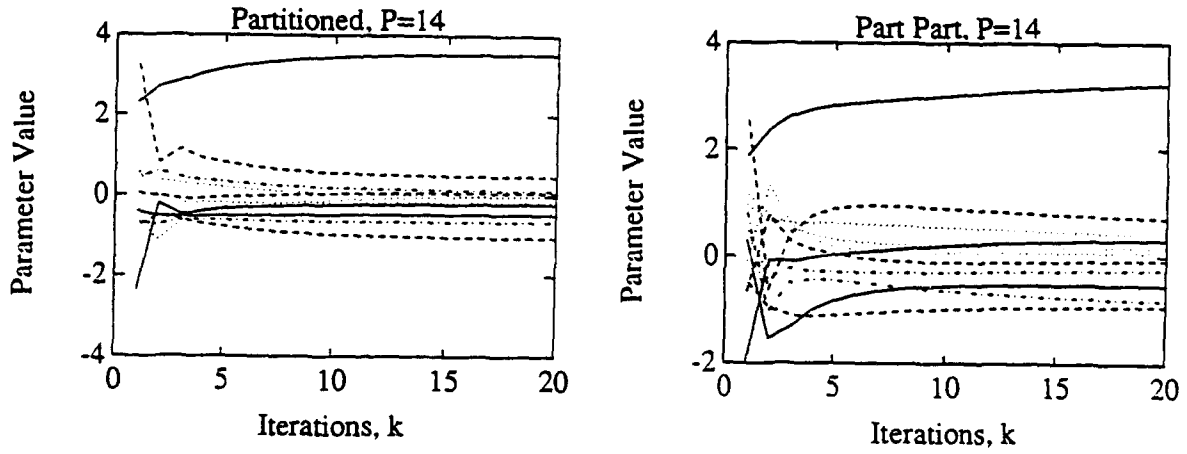


Figure 3.5: Parameter tracks and general rate of convergence for the FIR partitioned Toeplitz approximation and double-partitioned Toeplitz approximation algorithms.

real-time processing of signals as they are available is not possible. Data-adaptive algorithms compute a solution based upon the inclusion of new data with each iteration. They provide a means of providing a real-time system identification solution. In this section we develop a data-adaptive algorithm by extending the Toeplitz approximation technique presented in the previous section. Also, we present the well known RLS algorithm for comparison.

1. Toeplitz Approximation Algorithm

In order to avoid confusion with previous developments of the basic problem for the fixed data case, we define the following:

$$\mathbf{x}_n = [x(n) \quad x(n-1) \quad \cdots \quad x(n-M)]^T \quad (3.32)$$

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{bmatrix} \quad (3.33)$$

so that now $y(n) = \mathbf{x}_n^T \mathbf{a}$ for a specific value of the output of an M^{th} -order filter. For the general problem development, we have the following:

$$\begin{bmatrix} x(n) & x(n-1) & \cdots & x(n-M) \\ x(n-1) & x(n-2) & \cdots & x(n-M-1) \\ \vdots & \vdots & & \vdots \\ x(n-P) & x(n-P-1) & \cdots & x(n-P-M) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} y(n) \\ y(n-1) \\ \vdots \\ y(n-P) \end{bmatrix} \quad (3.34)$$

or, in vector notation we have:

$$\begin{bmatrix} \mathbf{x}_n^T \\ \mathbf{x}_{n-1}^T \\ \vdots \\ \mathbf{x}_{n-P}^T \end{bmatrix} \mathbf{a} = \mathbf{y}, \quad (3.35)$$

and now it is easy to see that the basic problem has the form:

$$\underbrace{\begin{bmatrix} \mathbf{x}_n & \mathbf{x}_{n-1} & \cdots & \mathbf{x}_{n-P} \end{bmatrix}}_R \begin{bmatrix} \mathbf{x}_n^T \\ \mathbf{x}_{n-1}^T \\ \vdots \\ \mathbf{x}_{n-P}^T \end{bmatrix} \mathbf{a} = \underbrace{\begin{bmatrix} \mathbf{x}_n & \mathbf{x}_{n-1} & \cdots & \mathbf{x}_{n-P} \end{bmatrix}}_{\mathbf{r}} \mathbf{y}. \quad (3.36)$$

It is important to realize how the data-adaptive algorithm can be derived from R and \mathbf{r} . Using a subscript n , we can express R and \mathbf{r} in time varying notation as:

$$\begin{aligned} R_n &= \sum_{j=n-P}^n \mathbf{x}_j \mathbf{x}_j^T \\ &= \underbrace{\mathbf{x}_{n-P} \mathbf{x}_{n-P}^T + \mathbf{x}_{n-P+1} \mathbf{x}_{n-P+1}^T + \cdots + \mathbf{x}_{n-1} \mathbf{x}_{n-1}^T}_{R_{n-1}} + \mathbf{x}_n \mathbf{x}_n^T \\ R_n &= R_{n-1} + \mathbf{x}_n \mathbf{x}_n^T \end{aligned} \quad (3.37)$$

and

$$\begin{aligned} \mathbf{r}_n &= \sum_{j=n-P}^n \mathbf{x}_j y(j) \\ &= \underbrace{\mathbf{x}_{n-P} y(n-P) + \mathbf{x}_{n-P+1} y(n-P+1) + \cdots + \mathbf{x}_{n-1} y(n-1)}_{\mathbf{r}_{n-1}} + \mathbf{x}_n y(n) \\ \mathbf{r}_n &= \mathbf{r}_{n-1} + \mathbf{x}_n y(n). \end{aligned} \quad (3.38)$$

Now that we have R_n and \mathbf{r}_n , we can form the data-adaptive Toeplitz approximation algorithm as in the fixed data case:

$$R_n \mathbf{a} = \mathbf{r}_n \quad (3.39)$$

substituting $R_n = T_n + S_n$ yields

$$(T_n + S_n)\mathbf{a} = \mathbf{r}_n, \quad (3.40)$$

and rearranging gives

$$T_n\mathbf{a} = \mathbf{r}_n - S_n\mathbf{a}. \quad (3.41)$$

But $S_n = R_n - T_n$, thus

$$T_n\mathbf{a} = \mathbf{r}_n - (R_n - T_n)\mathbf{a} \quad (3.42)$$

and isolating the parameter vector \mathbf{a} we have

$$\mathbf{a} = T_n^{-1}\mathbf{r}_n - T_n^{-1}R_n\mathbf{a} + \mathbf{a}. \quad (3.43)$$

Finally, the iterative algorithm has the form:

$$\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + T_n^{-1}\mathbf{r}_n - T_n^{-1}R_n\mathbf{a}^{(k)}. \quad (3.44)$$

This algorithm converges quickly to the true parameter values (see Fig 3.6). Notice that both algorithms begin with time $n = k = 0$. The data-adaptive algorithm begins to estimate parameter values immediately, but the fixed data algorithm must wait until a minimum of $M + 1$ data points have been accumulated before it can begin estimating parameter values for an M^{th} -order system. Although the data-adaptive algorithm (3.44) outperforms the fixed data algorithm of (3.31), there is a considerable computational tradeoff. The matrices T_n^{-1} and R_n , and the vector \mathbf{r}_n must be updated at each iteration of the data-adaptive algorithm. The matrix T_n^{-1} is updated using the matrix inversion lemma [Ref. 10], and R_n and \mathbf{r}_n are updated from (3.37) and (3.38), respectively. This requires approximately $5M^2$ additional multiplications for each iteration of (3.44) as compared with the fixed data algorithm.

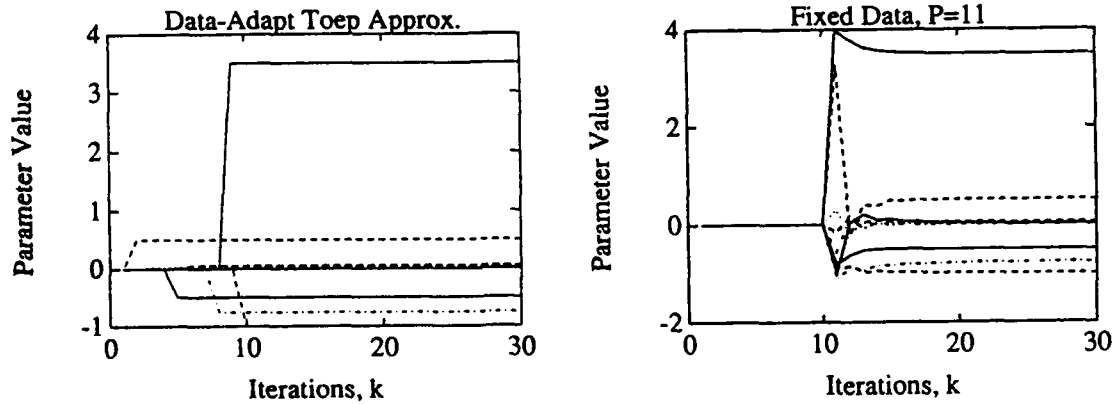


Figure 3.6: Parameter tracks and general rate of convergence for the FIR data-adaptive and fixed data Toeplitz approximation algorithms.

2. Recursive Least Squares (RLS) Algorithm

In order to evaluate the effectiveness of the data-adaptive Toeplitz approximation algorithm above, it must be compared against some benchmark. The RLS algorithm is recognized as a particularly optimal approach to the least squares FIR problem. A simple presentation of the update algorithm is [Ref. 5]:

$$\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \alpha R_{n+1}^{-1} \mathbf{x}_n e(n) \quad (3.45)$$

where α is referred to as the *forgetting factor*. A performance comparison between this algorithm and that of (3.44) reveals the astonishing fact that they perform exactly the same. Appendix A shows several examples of this under different conditions of noise, and over- and under-modeling. On close analysis, we note that the RLS algorithm (3.45) can be expanded and rewritten in the same form as (3.44). Although the algorithms look and perform the same, they are in fact different. Notice that R_n and \mathbf{r}_n are summed over the current and past data in (3.44), whereas these terms in the RLS algorithm are composed of only the current data values ($R_n = \mathbf{x}_n^T \mathbf{x}_n$ and

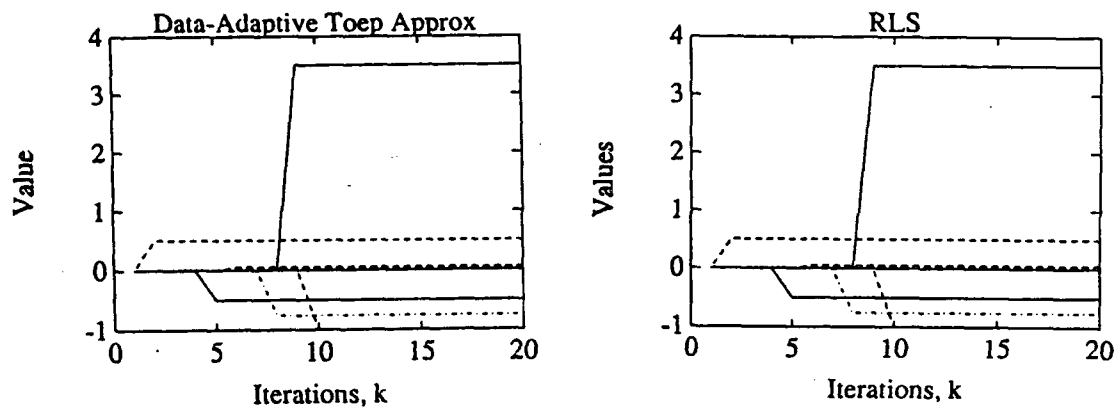


Figure 3.7: Parameter tracks and general rate of convergence for the FIR RLS and data-adaptive Toeplitz approximation algorithms.

$\mathbf{r}_n = \mathbf{x}_n y(n)$). Interestingly, as shown in Figure 3.7, the performance of the algorithms is still the same. Unfortunately, computing a more complicated \mathbf{R}_n and \mathbf{r}_n in the Toeplitz Approximation data-adaptive algorithm requires an increase of approximately $2M^2 - M$ multiplications over RLS. Although this increased computation does not produce any improvement in performance over FIR RLS systems modeling, it will give improved performance over IIR RLS, as we shall see in Chapter IV.

IV. INFINITE IMPULSE RESPONSE (IIR) SYSTEMS

In this chapter we consider the *infinite impulse response* (IIR) system. Here again we develop two general categories of solution algorithms, fixed data and data-adaptive. The slow rate of convergence of the Gauss-Seidel iterative algorithm applies to IIR systems in the same way it applied to FIR systems in Chapter III. Therefore, in this chapter we only present one fixed data algorithm, the Toeplitz approximation iterative method. The data-adaptive algorithms are the recursive least squares (RLS) method and Toeplitz approximation method. The same discussion regarding the use of techniques that are Toeplitz dependent from Chapter III is applicable here as well; our covariance formulation of the problem precludes us from using such techniques.

A. FIXED DATA ALGORITHMS

1. Toeplitz Approximation Algorithm

Here we present a Toeplitz approximation iterative algorithm [Ref. 2] of the same structure as in the FIR case. One major difference is that we will start with a partitioned version of the algorithm similar to that of (3.31). This is quite natural since we are solving for two parameter vectors. One vector, \mathbf{a}^M , represents the filter input or *feedforward* coefficients and the other, \mathbf{b}^N , represents the filter output or *feedback* coefficients. As in Chapter III, we drop the superscripts M and N . Unless otherwise stated, the input coefficient vector has order M , and the output coefficient vector has order N . Thus, from Chapter II, (2.18), we begin with

$$R\theta = \mathbf{r} \quad (4.1)$$

and partitioning R gives

$$\begin{bmatrix} R_{yy} & \vdots & R_{yx} \\ \dots\dots\dots \\ R_{xy} & \vdots & R_{xx} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \cdot \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_y \\ \cdot \\ \mathbf{r}_x \end{bmatrix}. \quad (4.2)$$

We now split the matrices R_{yy} and R_{xx} into a Toeplitz and a residual matrix:

$$\begin{bmatrix} T_{yy} + S_{yy} & \vdots & R_{yx} \\ \dots\dots\dots \\ R_{xy} & \vdots & T_{xx} + S_{xx} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \cdot \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_y \\ \cdot \\ \mathbf{r}_x \end{bmatrix}. \quad (4.3)$$

Multiplication gives the following coupled equations:

$$\begin{aligned} (T_{yy} + S_{yy})\mathbf{b} + R_{yx}\mathbf{a} &= \mathbf{r}_y \\ (T_{xx} + S_{xx})\mathbf{a} + R_{xy}\mathbf{b} &= \mathbf{r}_x \end{aligned} \quad (4.4)$$

and rearranging these gives

$$\begin{aligned} T_{yy}\mathbf{b} &= \mathbf{r}_y - R_{yx}\mathbf{a} - S_{yy}\mathbf{b} \\ T_{xx}\mathbf{a} &= \mathbf{r}_x - R_{xy}\mathbf{b} - S_{xx}\mathbf{a}. \end{aligned} \quad (4.5)$$

But $S_{yy} = R_{yy} - T_{yy}$ and $S_{xx} = R_{xx} - T_{xx}$, thus

$$\begin{aligned} \mathbf{b} &= T_{yy}^{-1}\mathbf{r}_y - T_{yy}^{-1}(R_{yx}\mathbf{a} + (R_{yy} - T_{yy})\mathbf{b}) \\ \mathbf{a} &= T_{xx}^{-1}\mathbf{r}_x - T_{xx}^{-1}(R_{xy}\mathbf{b} + (R_{xx} - T_{xx})\mathbf{a}) \end{aligned} \quad (4.6)$$

and letting $\mathbf{b}_0 = T^{-1}\mathbf{r}_y$ and $\mathbf{a}_0 = T^{-1}\mathbf{r}_x$ as the initial estimates of the parameter values, we have

$$\begin{aligned} \mathbf{b} &= \mathbf{b}_0 + \mathbf{b} - T_{yy}^{-1}(R_{yx}\mathbf{a} + R_{yy}\mathbf{b}) \\ \mathbf{a} &= \mathbf{a}_0 + \mathbf{a} - T_{xx}^{-1}(R_{xy}\mathbf{b} + R_{xx}\mathbf{a}). \end{aligned} \quad (4.7)$$

The final form of the fixed data IIR Toeplitz approximation iterative algorithm is:

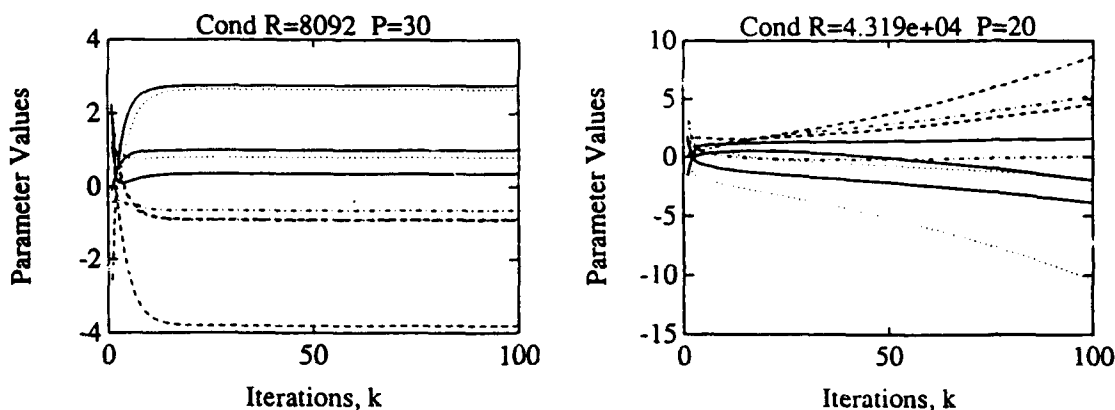


Figure 4.1: Parameter tracks and general rate of convergence for the IIR Toeplitz approximation algorithm.

$$\begin{aligned} \mathbf{b}^{(k+1)} &= \mathbf{b}_0 + \mathbf{b}^{(k)} - T_{yy}^{-1}(R_{yx}\mathbf{a}^{(k)} + R_{yy}\mathbf{b}^{(k)}) \\ \mathbf{a}^{(k+1)} &= \mathbf{a}_0 + \mathbf{a}^{(k)} - T_{xx}^{-1}(R_{xy}\mathbf{b}^{(k+1)} + R_{xx}\mathbf{a}^{(k)}). \end{aligned} \quad (4.8)$$

In order to evaluate the general performance of the algorithms developed in this chapter, the following IIR system is used for simulation:

$$\begin{aligned} y(n) &= 2.7600y(n-1) - 3.8090y(n-2) + 2.6540y(n-3) - 0.9240y(n-4) \\ &+ x(n) - 0.9x(n-1) + 0.81x(n-2) - 0.65x(n-3) + 0.36x(n-4). \end{aligned} \quad (4.9)$$

Figure 4.1 shows that this algorithm gives performance similar to that of the FIR algorithm (3.15). It converges relatively quickly when R is well-conditioned, and fails to converge to the true parameter values as R becomes ill-conditioned.

2. Toeplitz Approximation with Diagonal Perturbation Algorithm

As in the FIR case a significant shortcoming of (4.8) is that it fails to converge as the covariance matrix R becomes extremely ill-conditioned. We again observe that as the number of data points used to formulate R is reduced from a

large number (i.e., 5 to 10 times the order of the IIR filter being solved for) to near the minimum number required for a unique solution (i.e., the order of the filter), the condition number of R grows quite large. Under these circumstances the algorithm of (4.8) does not converge to the true parameter values. We now employ the diagonal perturbation technique in a manner similar to that of Chapter III which improves the condition of R and permits a modified version of (4.8) to converge as the number of data points approaches the minimum required.

If we let σ_y and σ_x be scalar multipliers and the Toeplitz approximation splitting of R_{yy} and R_{xx} be defined as $R_{yy} = T_{yy_0} + S_{yy}$ and $R_{xx} = T_{xx_0} + X_{xx}$, respectively, then we can derive a modified version of (4.8) as follows:

$$R\theta = r \quad (4.10)$$

adding a diagonal matrix to R_{yy} and R_{xx} gives,

$$\begin{bmatrix} R_{yy} + \sigma_y I & : & R_{yx} \\ \dots\dots\dots & & \dots\dots\dots \\ R_{xy} & : & R_{xx} + \sigma_x I \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \cdot \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_y \\ \cdot \\ \mathbf{r}_x \end{bmatrix} + \begin{bmatrix} \sigma_y \mathbf{b} \\ \dots \\ \sigma_x \mathbf{a} \end{bmatrix} \quad (4.11)$$

and substituting for R_{yy} and R_{xx} we have

$$\begin{bmatrix} T_{yy_0} + S_{yy} + \sigma_y I & : & R_{yx} \\ \dots\dots\dots & & \dots\dots\dots \\ R_{xy} & : & T_{xx_0} + S_{xx} + \sigma_x I \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \cdot \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_y \\ \cdot \\ \mathbf{r}_x \end{bmatrix} + \begin{bmatrix} \sigma_y \mathbf{b} \\ \dots \\ \sigma_x \mathbf{a} \end{bmatrix}. \quad (4.12)$$

Now, letting $T_{yy} = T_{yy_0} + \sigma_y I$ and $T_{xx} = T_{xx_0} + \sigma_x I$, and multiplying gives the following coupled equations:

$$\begin{aligned} (T_{yy} + S_{yy})\mathbf{b} + R_{yx}\mathbf{a} &= \mathbf{r}_y + \sigma_y \mathbf{b} \\ (T_{xx} + S_{xx})\mathbf{a} + R_{xy}\mathbf{b} &= \mathbf{r}_x + \sigma_x \mathbf{a} \end{aligned} \quad (4.13)$$

and rearranging these gives

$$\begin{aligned}
T_{yy}\mathbf{b} &= \mathbf{r}_y + \sigma_y\mathbf{b} - R_{yx}\mathbf{a} - S_{yy}\mathbf{b} \\
T_{xx}\mathbf{a} &= \mathbf{r}_x + \sigma_x\mathbf{a} - R_{xy}\mathbf{b} - S_{xx}\mathbf{a}.
\end{aligned} \tag{4.14}$$

Substituting for $S_{yy} = R_{yy} + \sigma_y I - T_{yy}$ and $S_{xx} = R_{xx} + \sigma_x I - T_{xx}$ we have

$$\begin{aligned}
T_{yy}\mathbf{b} &= \mathbf{r}_y + \sigma_y\mathbf{b} - R_{yx}\mathbf{a} - (R_{yy} + \sigma_y I - T_{yy})\mathbf{b} \\
T_{xx}\mathbf{a} &= \mathbf{r}_x + \sigma_x\mathbf{a} - R_{xy}\mathbf{b} - (R_{xx} + \sigma_x I - T_{xx})\mathbf{a},
\end{aligned} \tag{4.15}$$

and now we see the terms $\sigma_y\mathbf{b}$ and $\sigma_x\mathbf{a}$ cancel out:

$$\begin{aligned}
T_{yy}\mathbf{b} &= \mathbf{r}_y + \sigma_y\mathbf{b} - \sigma_y\mathbf{b} - R_{yx}\mathbf{a} - (R_{yy} - T_{yy})\mathbf{b} \\
T_{xx}\mathbf{a} &= \mathbf{r}_x + \sigma_x\mathbf{a} - \sigma_x\mathbf{a} - R_{xy}\mathbf{b} - (R_{xx} - T_{xx})\mathbf{a}.
\end{aligned} \tag{4.16}$$

After simplifying to isolate the parameter vectors, we have

$$\begin{aligned}
\mathbf{b} &= T_{yy}^{-1}\mathbf{r}_y - T_{yy}^{-1}(R_{yx}\mathbf{a} + (R_{yy} - T_{yy})\mathbf{b}) \\
\mathbf{a} &= T_{xx}^{-1}\mathbf{r}_x - T_{xx}^{-1}(R_{xy}\mathbf{b} + (R_{xx} - T_{xx})\mathbf{a}).
\end{aligned} \tag{4.17}$$

Again, letting $T_{yy}^{-1}\mathbf{r}_y = \mathbf{b}_0$ and $T_{xx}^{-1}\mathbf{r}_x = \mathbf{a}_0$ gives

$$\begin{aligned}
\mathbf{b} &= \mathbf{b}_0 + \mathbf{b} - T_{yy}^{-1}(R_{yx}\mathbf{a} + R_{yy}\mathbf{b}) \\
\mathbf{a} &= \mathbf{a}_0 + \mathbf{a} - T_{xx}^{-1}(R_{xy}\mathbf{b} + R_{xx}\mathbf{a})
\end{aligned} \tag{4.18}$$

and the final form of the Toeplitz approximation with diagonal perturbation iterative algorithm is:

$$\begin{aligned}
\mathbf{b}^{(k+1)} &= \mathbf{b}_0 + \mathbf{b}^{(k)} - T_{yy}^{-1}(R_{yx}\mathbf{a}^{(k)} + R_{yy}\mathbf{b}^{(k)}) \\
\mathbf{a}^{(k+1)} &= \mathbf{a}_0 + \mathbf{a}^{(k)} - T_{xx}^{-1}(R_{xy}\mathbf{b}^{(k+1)} + R_{xx}\mathbf{a}^{(k)}).
\end{aligned} \tag{4.19}$$

The algorithm of (4.19) has exactly the same form as (4.8). Unfortunately, adding a diagonal value dramatically slows down the rate of convergence; however, the new algorithm (4.19) was observed to converge where the algorithm (4.8) fails. This

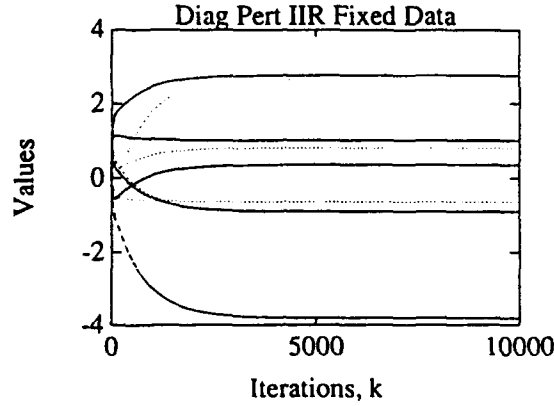


Figure 4.2: Parameter tracks and general rate of convergence for the IIR Toeplitz approximation with diagonal perturbation algorithm.

algorithm takes a large number (i.e., several hundreds or thousands) of iterations just to reach the vicinity of the true parameter values, and still requires a large number of further iterations to converge to the correct solution. We did observe that appropriate values for σ_y and σ_x are inversely proportional to the condition number of R_{yy} and R_{xx} , respectively. Unfortunately, even the optimal performance of this algorithm is unsatisfactory (see Figure 4.2).

3. Partitioned Toeplitz Approximation Algorithm

In this section we block partition R_{yy} and R_{xx} in an attempt to improve the performance of (4.8). As before, the number of row partitions must equal the number of column partitions. Similar to previous developments we now derive a partitioned version of (4.8). Beginning with

$$R\theta = r, \quad (4.20)$$

partitioning twice gives:

$$\begin{bmatrix} \left(\begin{array}{c|c} R_{yy11} & R_{yy12} \\ \hline R_{yy21} & R_{yy22} \end{array} \right) & \left(\begin{array}{c|c} R_{yx11} & R_{yx12} \\ \hline R_{yx21} & R_{yx22} \end{array} \right) \\ \hline \left(\begin{array}{c|c} R_{xy11} & R_{xy12} \\ \hline R_{xy21} & R_{xy22} \end{array} \right) & \left(\begin{array}{c|c} R_{xx11} & R_{xx12} \\ \hline R_{xx21} & R_{xx22} \end{array} \right) \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \hline \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{y1} \\ \mathbf{r}_{y2} \\ \hline \mathbf{r}_{x1} \\ \mathbf{r}_{x2} \end{bmatrix}. \quad (4.21)$$

Splitting the diagonal blocks into a Toeplitz and a residual matrix we have

$$\begin{bmatrix} \left(\begin{array}{c|c} T_{yy11} + S_{yy11} & R_{yy12} \\ \hline R_{yy21} & T_{yy22} + S_{yy22} \end{array} \right) & \left(\begin{array}{c|c} R_{yx11} & R_{yx12} \\ \hline R_{yx21} & R_{yx22} \end{array} \right) \\ \hline \left(\begin{array}{c|c} R_{xy11} & R_{xy12} \\ \hline R_{xy21} & R_{xy22} \end{array} \right) & \left(\begin{array}{c|c} T_{xx11} + S_{xx11} & R_{xx12} \\ \hline R_{xx21} & T_{xx22} + S_{xx22} \end{array} \right) \end{bmatrix} \times \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \hline \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{y1} \\ \mathbf{r}_{y2} \\ \hline \mathbf{r}_{x1} \\ \mathbf{r}_{x2} \end{bmatrix}. \quad (4.22)$$

Multiplying gives four coupled equations:

$$\begin{aligned} (T_{yy11} + S_{yy11})\mathbf{b}_1 + R_{yy12}\mathbf{b}_2 + R_{yx11}\mathbf{a}_1 + R_{yx12}\mathbf{a}_2 &= \mathbf{r}_{y1} \\ (T_{yy22} + S_{yy22})\mathbf{b}_2 + R_{yy21}\mathbf{b}_1 + R_{yx21}\mathbf{a}_1 + R_{yx22}\mathbf{a}_2 &= \mathbf{r}_{y2} \\ (T_{xx11} + S_{xx11})\mathbf{a}_1 + R_{xy11}\mathbf{b}_1 + R_{xy12}\mathbf{b}_2 + R_{xx12}\mathbf{a}_2 &= \mathbf{r}_{x1} \\ (T_{xx22} + S_{xx22})\mathbf{a}_2 + R_{xy21}\mathbf{b}_1 + R_{xy22}\mathbf{b}_2 + R_{xx21}\mathbf{a}_1 &= \mathbf{r}_{x2} \end{aligned} \quad (4.23)$$

and rearranging yields

$$\begin{aligned}
T_{yy11}b_1 &= r_{y1} - R_{yy12}b_2 - R_{yx11}a_1 - R_{yx12}a_2 - S_{yy11}b_1 \\
T_{yy22}b_2 &= r_{y2} - R_{yy21}b_1 - R_{yx21}a_1 - R_{yx22}a_2 - S_{yy22}b_2 \\
T_{xx11}a_1 &= r_{x1} - R_{xy11}b_1 - R_{xy12}b_2 - R_{xx12}a_2 - S_{xx11}a_1 \\
T_{xx22}a_2 &= r_{x2} - R_{xy21}b_1 - R_{xy22}b_2 - R_{xx21}a_1 - S_{xx22}a_2.
\end{aligned} \tag{4.24}$$

Now, isolating the parameter vectors and substituting for $S_{yy11} = R_{yy11} - T_{yy11}$, etc.,

$$\begin{aligned}
b_1 &= T_{yy11}^{-1}r_{y1} - T_{yy11}^{-1}(R_{yy12}b_2 - R_{yx11}a_1 - R_{yx12}a_2 - (R_{yy11} - T_{yy11})b_1) \\
b_2 &= T_{yy22}^{-1}r_{y2} - T_{yy22}^{-1}(R_{yy21}b_1 - R_{yx21}a_1 - R_{yx22}a_2 - (R_{yy22} - T_{yy22})b_2) \\
a_1 &= T_{xx11}^{-1}r_{x1} - T_{xx11}^{-1}(R_{xy11}b_1 - R_{xy12}b_2 - R_{xx12}a_2 - (R_{xx11} - T_{xx11})a_1) \\
a_2 &= T_{xx22}^{-1}r_{x2} - T_{xx22}^{-1}(R_{xy21}b_1 - R_{xy22}b_2 - R_{xx21}a_1 - (R_{xx22} - T_{xx22})a_2)
\end{aligned} \tag{4.25}$$

and letting $b_{10} = T_{yy11}^{-1}r_{y1}$, etc., as the initial estimate of the parameter vectors, we have

$$\begin{aligned}
b_1 &= b_{10} + b_1 - T_{yy11}^{-1}(R_{yy12}b_2 - R_{yx11}a_1 - R_{yx12}a_2 - R_{yy11}b_1) \\
b_2 &= b_{20} + b_2 - T_{yy22}^{-1}(R_{yy21}b_1 - R_{yx21}a_1 - R_{yx22}a_2 - R_{yy22}b_2) \\
a_1 &= a_{10} + a_1 - T_{xx11}^{-1}(R_{xy11}b_1 - R_{xy12}b_2 - R_{xx12}a_2 - R_{xx11}a_1) \\
a_2 &= a_{20} + a_2 - T_{xx22}^{-1}(R_{xy21}b_1 - R_{xy22}b_2 - R_{xx21}a_1 - R_{xx22}a_2).
\end{aligned} \tag{4.26}$$

The final form of the partitioned fixed data Toeplitz approximation iterative algorithm is:

$$\begin{aligned}
b_1^{(k+1)} &= b_{10} + b_1^{(k)} - T_{yy11}^{-1}(R_{yy12}b_2^{(k)} - R_{yx11}a_1^{(k)} - R_{yx12}a_2^{(k)} - R_{yy11}b_1^{(k)}) \\
b_2^{(k+1)} &= b_{20} + b_2^{(k)} - T_{yy22}^{-1}(R_{yy21}b_1^{(k+1)} - R_{yx21}a_1^{(k)} - R_{yx22}a_2^{(k)} - R_{yy22}b_2^{(k)}) \\
a_1^{(k+1)} &= a_{10} + a_1^{(k)} - T_{xx11}^{-1}(R_{xy11}b_1^{(k+1)} - R_{xy12}b_2^{(k+1)} - R_{xx12}a_2^{(k)} - R_{xx11}a_1^{(k)}) \\
a_2^{(k+1)} &= a_{20} + a_2^{(k)} - T_{xx22}^{-1}(R_{xy21}b_1^{(k+1)} - R_{xy22}b_2^{(k+1)} - R_{xx21}a_1^{(k+1)} - R_{xx22}a_2^{(k)}).
\end{aligned} \tag{4.27}$$

This partitioned Toeplitz approximation algorithm was observed to converge to the true solution where the algorithm (4.8) fails, however it has the same

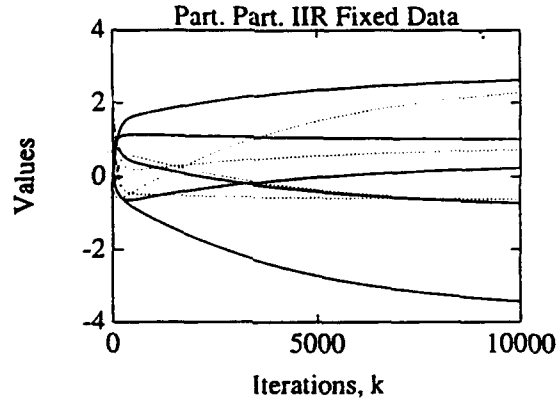


Figure 4.3: Parameter tracks and general rate of convergence for the IIR partitioned Toeplitz approximation algorithm.

problem as the diagonal perturbation technique. It takes an excessive number of iterations for the algorithm to approach the vicinity of the true parameter values and even longer to converge exactly (see Figure 4.3). Since this is not acceptable performance (even marginally), we therefore turn to data-adaptive techniques in order to achieve the desired result.

B. DATA-ADAPTIVE ALGORITHMS

1. Toeplitz Approximation Algorithm

Extending the development from equations (4.1)–(4.6) to the time-varying case gives the desired data-adaptive IIR Toeplitz approximation algorithm:

$$\begin{aligned} \mathbf{b}^{(k+1)} &= \mathbf{b}^{(k)} + T_{yy_n}^{-1}(\mathbf{r}_{y_n} - R_{yx_n}\mathbf{a}^{(k)} - R_{yy_n}\mathbf{b}^{(k)}) \\ \mathbf{a}^{(k+1)} &= \mathbf{a}^{(k)} + T_{xx_n}^{-1}(\mathbf{r}_{x_n} - R_{xy_n}\mathbf{b}^{(k)} - R_{xx_n}\mathbf{a}^{(k)}) \end{aligned} \quad (4.28)$$

where the subscript n denotes *time variation*. The neat and clean appearance of this algorithm is deceptive. Each of the terms with a subscript n must be updated

during each iteration. Consequently, this algorithm is somewhat computationally intensive. The entire algorithm is coded in matlab and is listed in Appendix D. It uses the matrix inversion lemma to compute the T_{yy_n} and T_{xx_n} updates. Updating the rest of the time varying terms is similar to the process shown in (3.37) and (3.38). We observed that the algorithm gave better results for over-modeling than exact modeling. An example of over-modeling is where the true system output was generated from a 4th-order system ($M = N = 4$), but we are using a 6th-order model ($M = N = 6$) in the algorithm. The results for this case are shown in Figure 4.4. Here we see that the parameter values converge in about 15 to 20 iterations. To see whether this 6th-order model faithfully represents the true 4th-order system, we must look at the frequency response of the model versus the true system. Figure 4.5 shows that by about 20 iterations the 6th-order model matches the frequency response of the true system. This performance is quite reasonable, and Appendix C contains parameter track and frequency response plots showing its performance under a variety of conditions. Another important point about this algorithm is that it does not use the gradient terms that appear in (2.21). In fact, when the gradient terms were incorporated in the algorithm, the performance was observed to be worse.

2. Recursive Least Squares (RLS) Algorithm

Unfortunately, there is not a well-defined algorithm that will serve as a benchmark against which to test the data-adaptive Toeplitz approximation algorithm. In this section we use, without derivation, the IIR RLS algorithm defined by [Ref. 5]:

$$\theta^{(k+1)} = \theta^{(k)} + \alpha R_{n+1}^{-1} \psi_n e(n) \quad (4.29)$$

where $\theta = [\mathbf{b}^T \mathbf{a}^T]^T$ is the parameter vector, $\psi_n = [\beta_n^T \alpha_n^T]^T$ is the gradient term vector, and α is the *forgetting factor*. There are some real difficulties in keeping this algorithm stable [Ref. 5], and it performs best when used to exactly model a system's

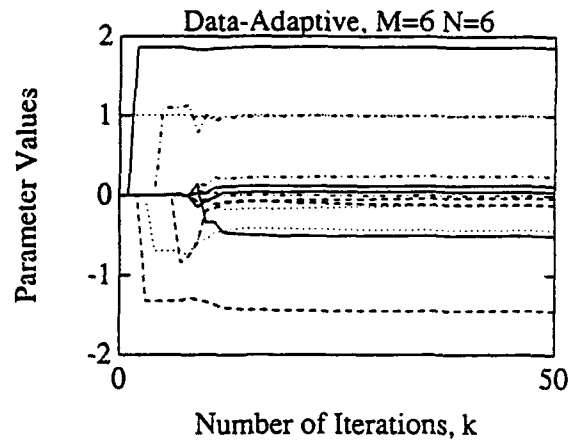


Figure 4.4: Parameter tracks and general rate of convergence for the IIR data-adaptive Toeplitz approximation algorithm.

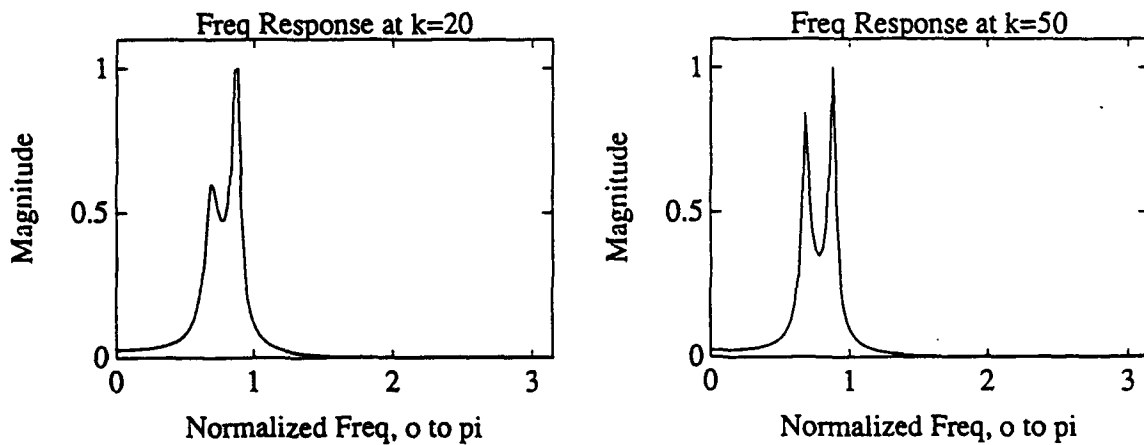


Figure 4.5: Frequency response of the Toeplitz approximation algorithm compared to the true system frequency response.

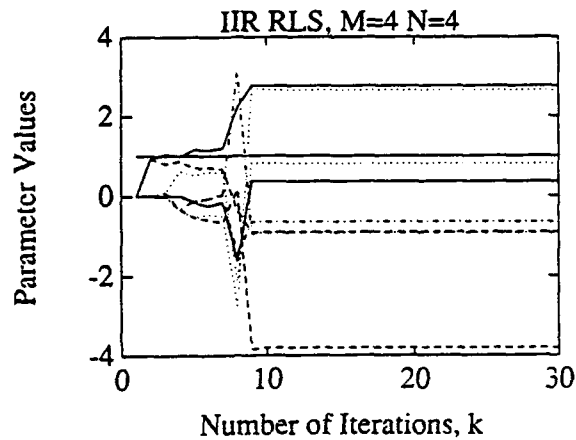


Figure 4.6: Parameter tracks and general rate of convergence for the IIR RLS algorithm.

parameters. The performance of this algorithm, when it is stable, is shown in Figure 4.6. However, for general system modeling where the exact order of the system is unknown, IIR RLS is unsatisfactory. See Figures C.10 thru C.14 of Appendix C.

Comparing the performance of IIR RLS to that of data-adaptive IIR Toeplitz approximation, we observe that the increased computations required in the Toeplitz approximation algorithm give improved performance. The stability of the Toeplitz approximation algorithm is clearly evident when we consider the poles and zeros of the model system as it converges. In Figure 4.7 the poles (denoted by x) and zeros (denoted by o) move inside the unit circle as the number of iterations goes from 9 to 20. The fact that the poles and zeros stop moving as the algorithm continues to iterate clearly demonstrates that it is stable. Also observe that the two additional poles and zeros overlap at iteration $k = 50$, which effectively cancels their contribution to the frequency response.

Similar performance is observed in the presence of noise, and the Toeplitz

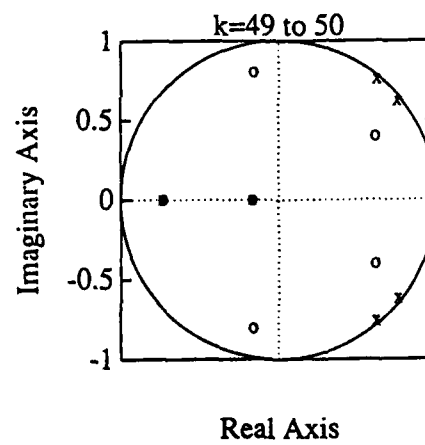
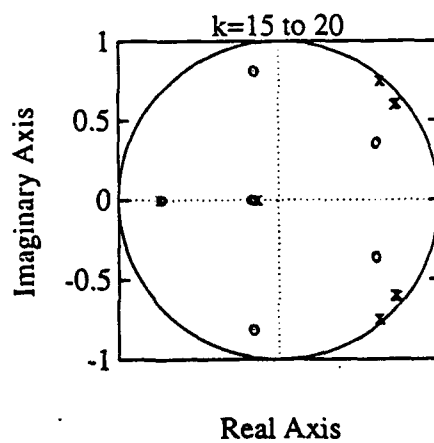
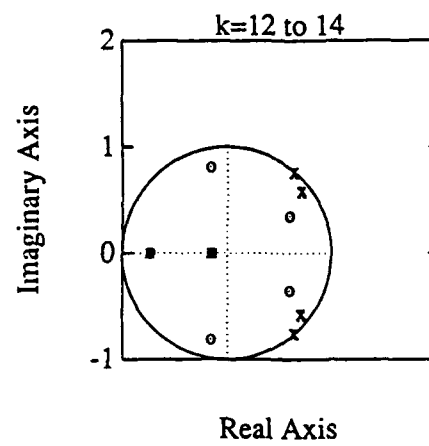
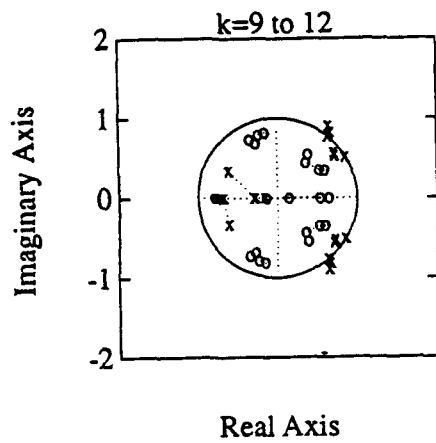


Figure 4.7: Pole-zero plots for the data-adaptive IIR Toeplitz approximation algorithm with $M = N = 6$.

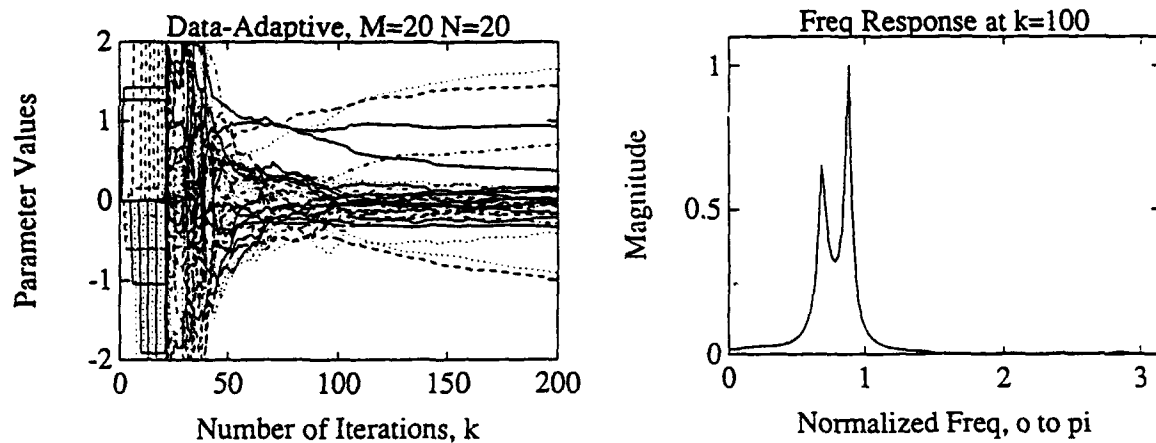


Figure 4.8: Parameter track and frequency response for the data-adaptive IIR Toeplitz approximation algorithm with $M = N = 20$ and 10 dB additive noise.

approximation algorithm converges faster to the true frequency response when higher order model systems are computed. Notice how the true frequency response emerges as the algorithm begins to converge for a $M = N = 20$ model system in the presence of 10 dB additive noise (see Figure 4.9).

V. CONCLUSIONS

In this thesis we applied the four concepts identified in the Introduction to both FIR and IIR system parameter estimation. Of these, diagonal perturbation and multiple partitioning are used to extend the basic fixed data algorithms of [Ref. 1] for FIR systems and of [Ref. 2] for IIR systems. We then used all four concepts to create new data-adaptive algorithms which extend the fixed data algorithms into the data-adaptive domain. In the following sections we draw general conclusions about the performance of these algorithms.

A. FIR ALGORITHMS

Improving the condition of the covariance matrix with increased diagonal dominance produced improved performance over the original fixed data FIR Toeplitz approximation algorithm. The algorithm converges fairly rapidly while using the minimum amount of data. Figures A.1 thru A.5 of Appendix A show the general performance of this algorithm in noise, as well as for over-modeling and under-modeling. The major drawback of this algorithm is the problem of determining a suitable value of the diagonal perturbation parameter σ that will give optimal performance.

Partitioning the covariance matrix also produced improved performance of the original fixed data FIR Toeplitz approximation algorithm. Using the minimum amount of data, the partitioned Toeplitz approximation algorithm converges almost as rapidly as the optimal case of the diagonal perturbation Toeplitz approximation algorithm. Figures A.6 thru A.9 of Appendix A show the general performance of this algorithm in noise, as well as for over-modeling and under-modeling. Significant features of this algorithm are that it does not require choosing an optimizing parameter, and the

partitioning reduces the computational intensity of the algorithm since the matrices involved are smaller.

Extending the original fixed data FIR Toeplitz approximation algorithm to the data-adaptive case gives an algorithm that performs quite well, equal to RLS, however it is more computationally intensive than RLS. Establishing this data-adaptive FIR algorithm is important because it demonstrates the potential for a data-adaptive IIR version to work as well. Figures A.10 thru A.17 of Appendix A show the general performance of this algorithm and RLS under different conditions of noise, as well as for over-modeling and under-modeling.

B. IIR ALGORITHMS

Applying the same diagonal perturbation and partitioning techniques used on the fixed data FIR Toeplitz approximation algorithm to the IIR algorithm did not produce the same improvement. Although these techniques did allow the algorithm to converge with the desired minimum amount of data, the rate of convergence is unacceptable. Figures B.1 thru B.9 of Appendix B show the general performance of these algorithms under different conditions of data and noise, as well as for over-modeling and under-modeling.

The data-adaptive IIR Toeplitz approximation algorithm succeeds in parameter estimation where the corresponding IIR RLS algorithms fail. It performs well under a wide range of noise and over-modeling conditions. We observed that even in moderate noise (i.e., 10 to 20 dB) over-modeling with this algorithm can be used to determine the true frequency response of the system being modeled. Figures C.1 thru C.14 of Appendix C show how the algorithms perform under different conditions of noise, as well as over-modeling and under-modeling. A significant consideration with this algorithm is the large amount of computations required for each iteration.

C. FUTURE WORK

Here we have reported general observations rather than a rigorous analytical development on the performance of the algorithms presented in the thesis. As future work, the results and conclusions from this work should be put on solid analytical footing. Also, the algorithms presented in this thesis are developed for one-dimensional real data only. The algorithms may be extended to incorporate complex and multi-dimensional data and related applications.

APPENDIX A: FIR SYSTEM SIMULATIONS

This appendix contains plots corresponding to the FIR algorithms developed in Chapter III. Figure A.1 shows the performance of the unmodified Toeplitz approximation algorithm (3.15). This plot also serves as a reference of the true system (3.8) frequency response. The remainder of the plots, Figures A.2-A.17, are in the following general order: fixed data Toeplitz approximation with diagonal perturbation, fixed data partitioned Toeplitz approximation, data-adaptive Toeplitz approximation, and finally FIR RLS.

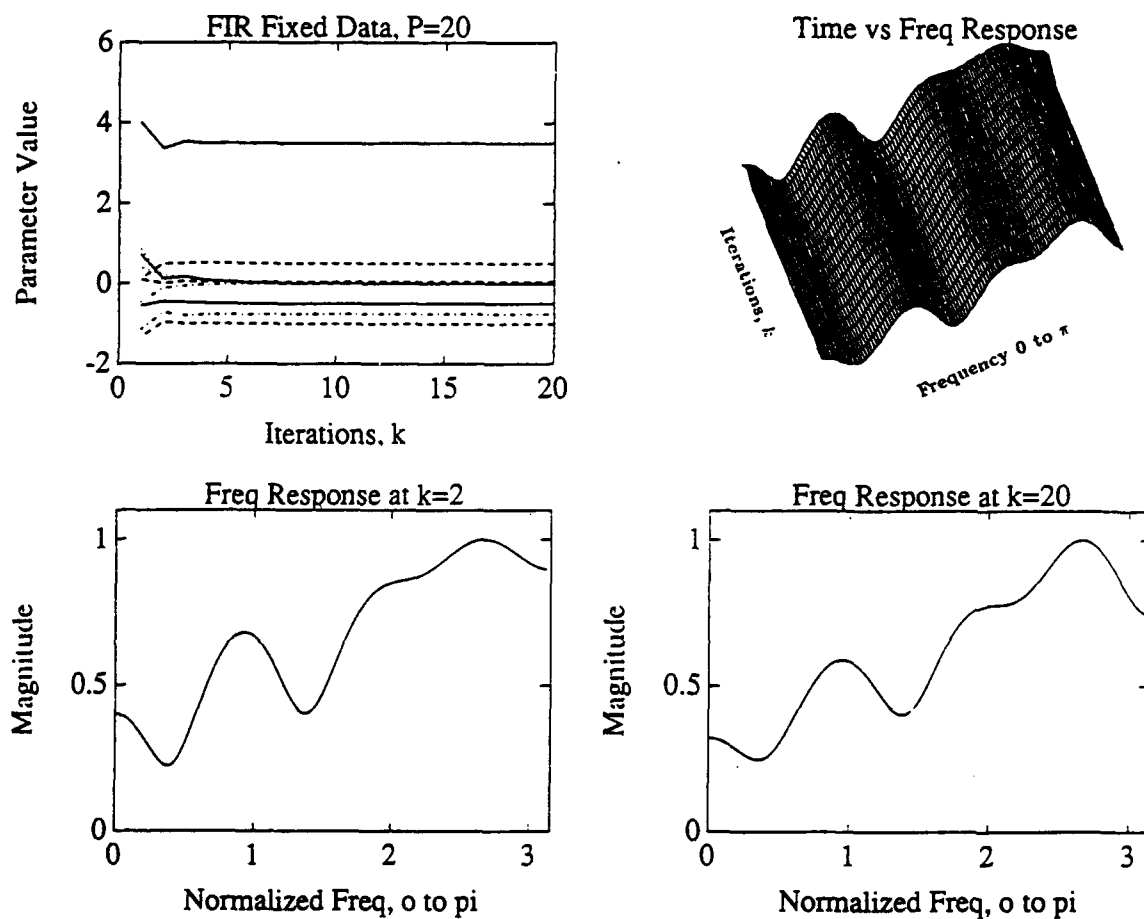


Figure A.1: The basic fixed data FIR Toeplitz approximation algorithm. This plot shows the parameter tracks and frequency response of the basic Toeplitz approximation algorithm (3.15). The correlation matrix was formed with $P = 20$ data points, and there was no additive noise in the output. The algorithm converges rapidly under these conditions, as can be seen by the flat parameter tracks. Also notice that the frequency response changes little as the number of iterations increase. The frequency response plot for $k = 20$ iterations represents the frequency response of the true system.

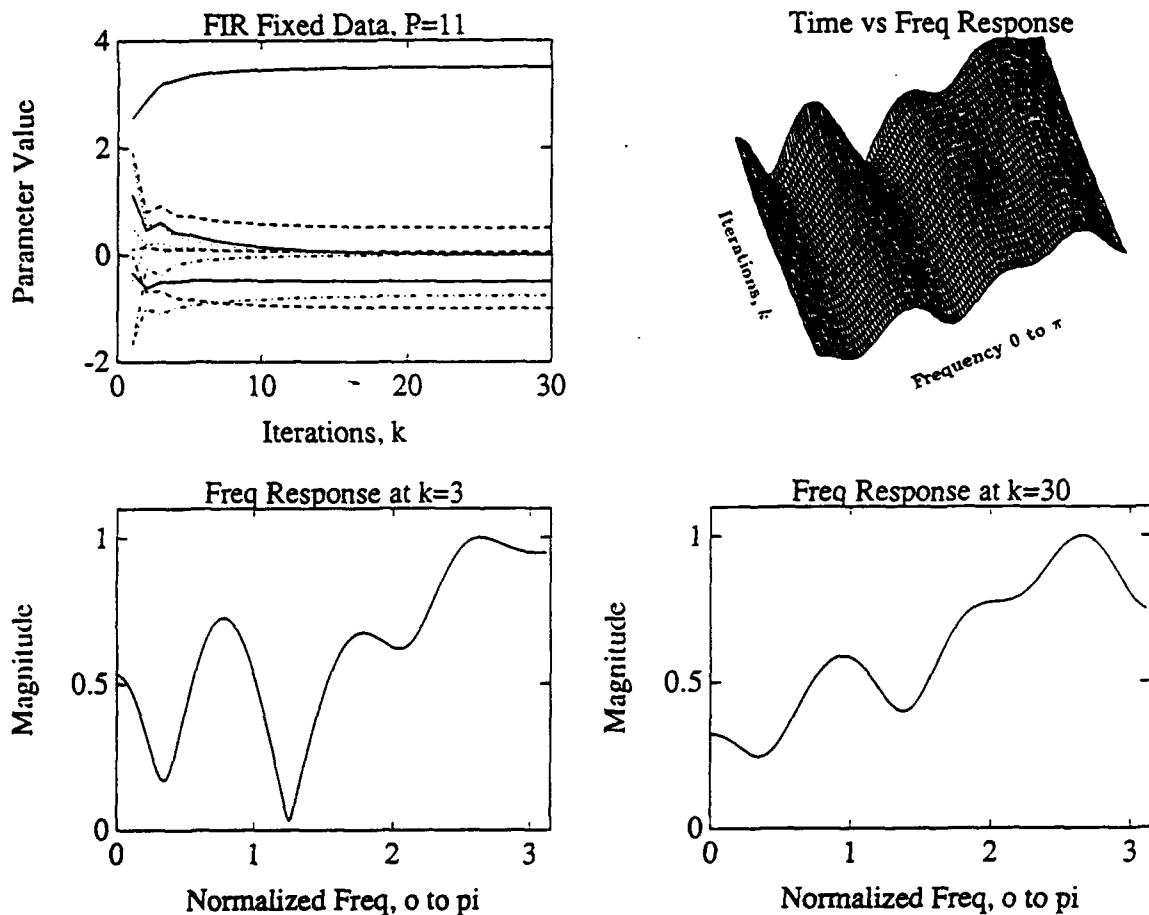


Figure A.2: Parameter tracks and frequency response of the fixed data Toeplitz approximation with diagonal perturbation algorithm.

Observe that even though the minimum number of data points ($P = 11$) were used to form the correlation matrix, the algorithm converges rather quickly. The frequency response plot for $k = 30$ iterations appears identical to the response of the true system shown in Figure A.1. No noise was added to the output, the perturbation parameter was $\sigma = 1$, and the true system was exactly modeled with a 10^{th} -order model.

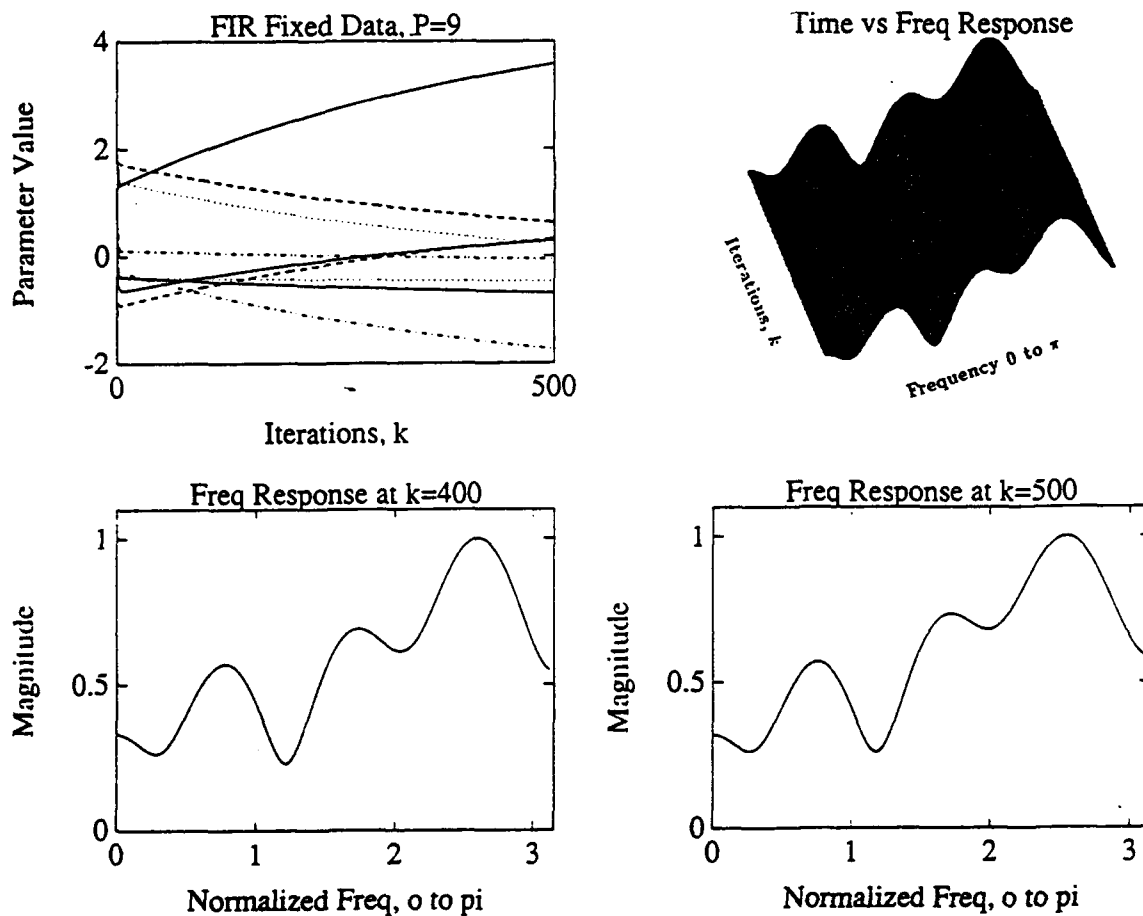


Figure A.3: Under-modeled example of the fixed data Toeplitz approximation with diagonal perturbation algorithm.

The 10th-order true system was modeled with an 8th-order model. The algorithm does a fairly good job preserving the frequency components of the true system, however it takes a much greater number of iterations to achieve this result. No noise was added to the output, and the perturbation parameter was $\sigma = 1$.

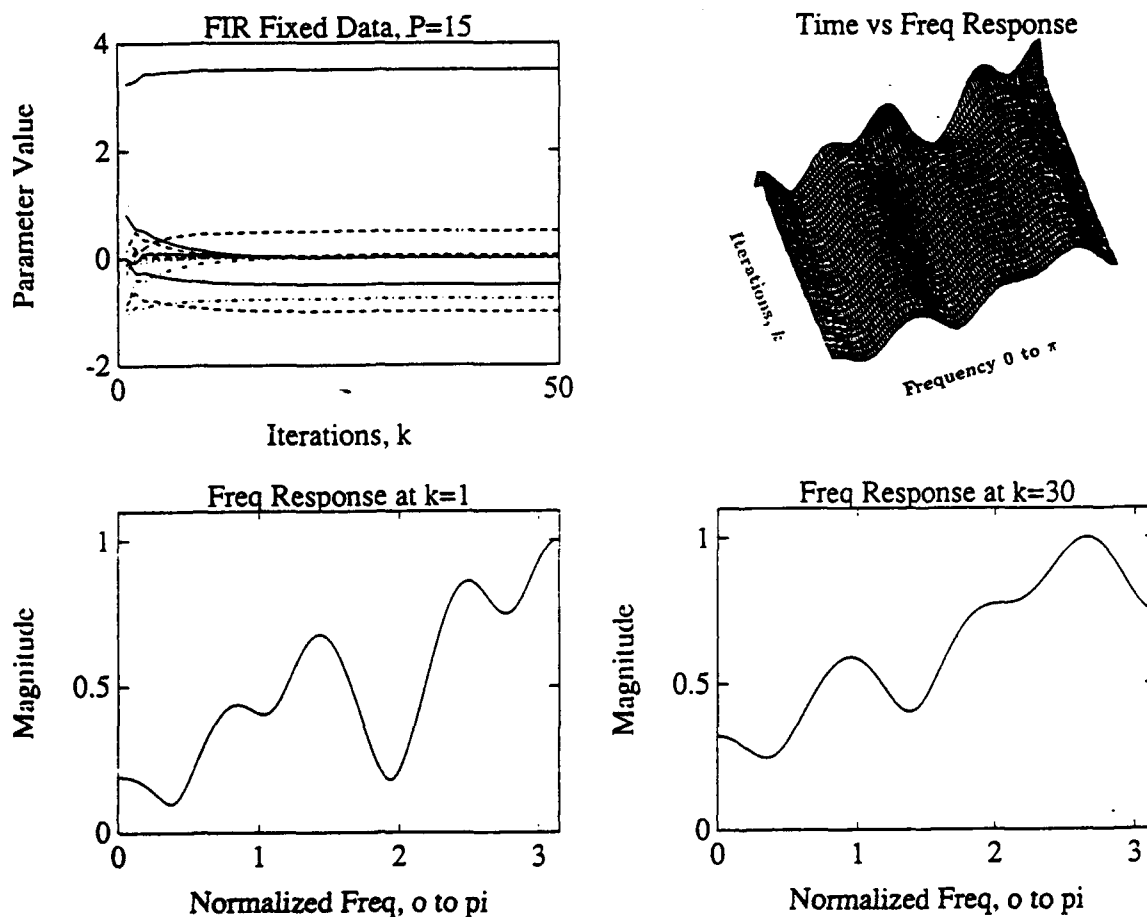


Figure A.4: Over-modeled example of the fixed data Toeplitz approximation with diagonal pertubation algorithm.

The 10th-order true system was modeled with a 14th-order model. The algorithm converges rapidly, and by $k = 30$ iterations it matches the true system frequency response. It is important to note that this result was achieved with the minimum ($P = 15$) amount of data required. No noise was added to the output, and the pertubation parameter was $\sigma = 3$.

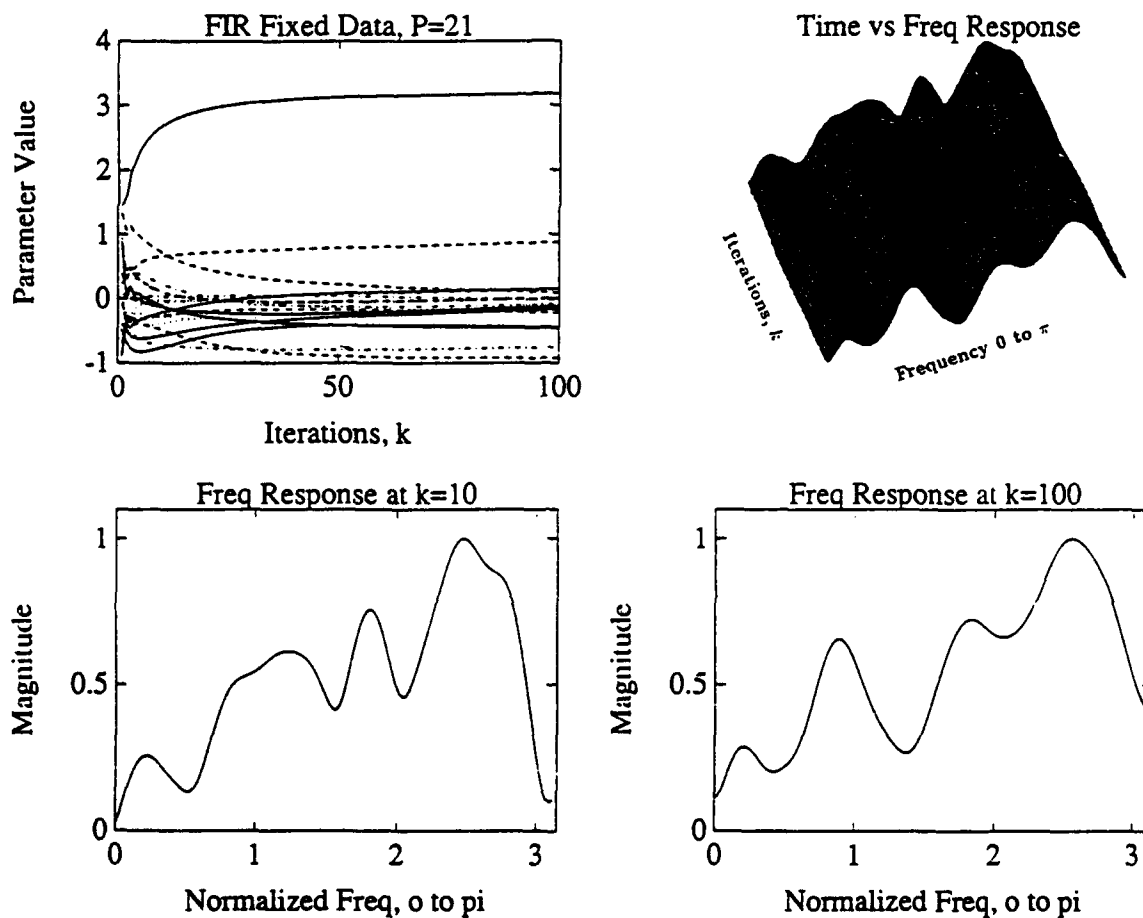


Figure A.5: Over-modeled example with 10 dB of additive noise for the fixed data Toeplitz approximation with diagonal perturbation algorithm. The 10th-order true system was modeled with a 20th-order model. The additive white noise tends to slow down the algorithm, and distorts the frequency response it is able to achieve. The perturbation parameter was $\sigma = 3$, and 10 dB of white noise was added to the output.

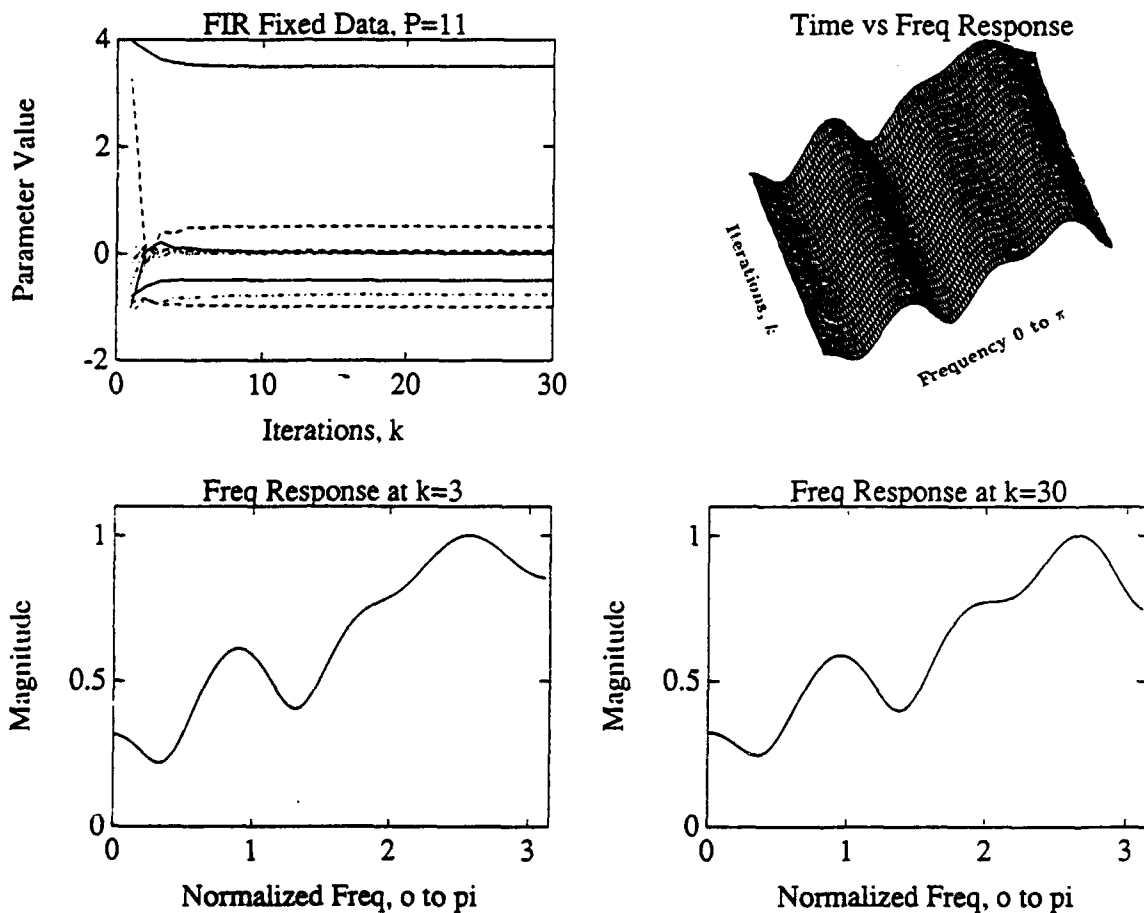


Figure A.6: Parameter tracks and frequency response of the fixed data partitioned Toeplitz approximation algorithm.

Observe that even though the minimum number of data points ($P = 11$) were used to form the correlation matrix, the algorithm converges rather quickly. The frequency response plot for $k = 30$ iterations appears identical to the response of the true system shown in Figure A.1. No noise was added to the output, and the true system was exactly modeled with a 10^{th} -order model.

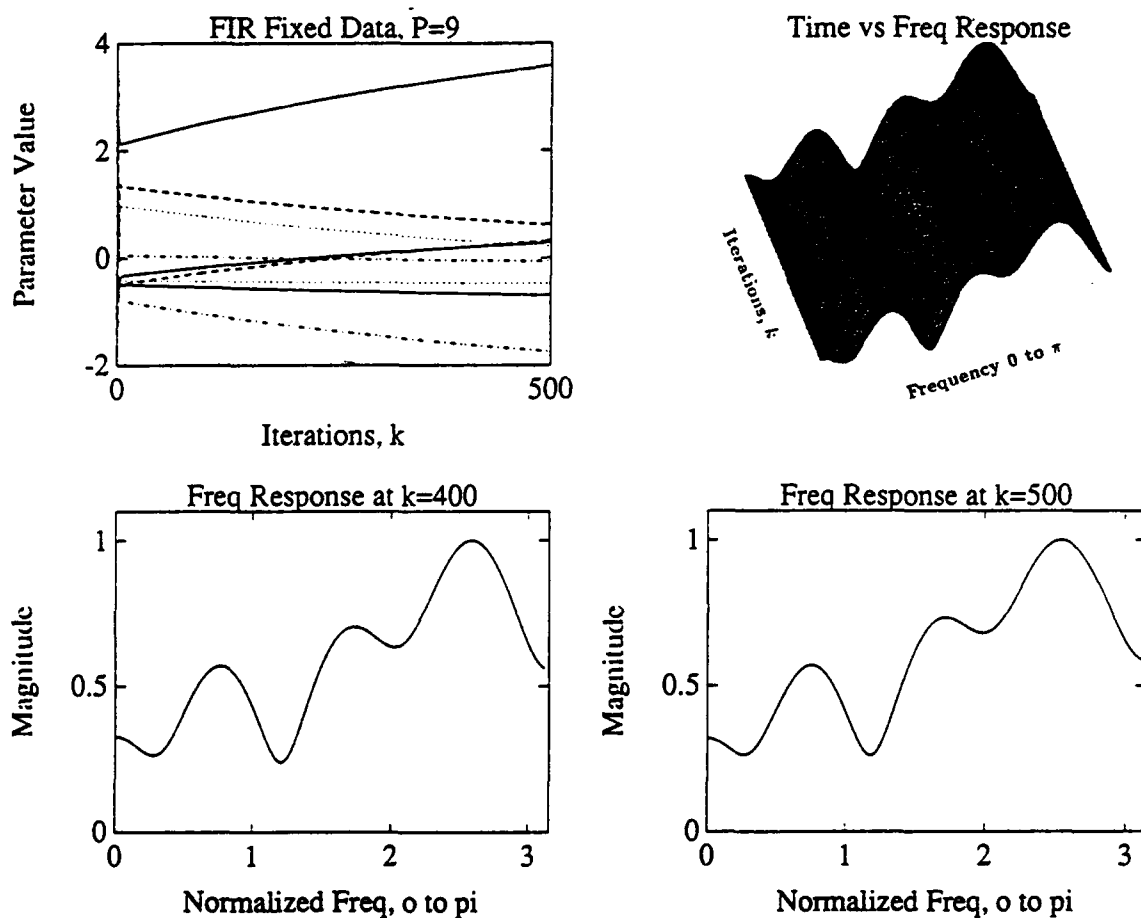


Figure A.7: Under-modeled example of the fixed data partitioned Toeplitz approximation algorithm.

The 10th-order true system was modeled with an 8th-order model. The algorithm does a fairly good job preserving the frequency components of the true system, however it takes a much greater number of iterations to achieve this result. No noise was added to the output.

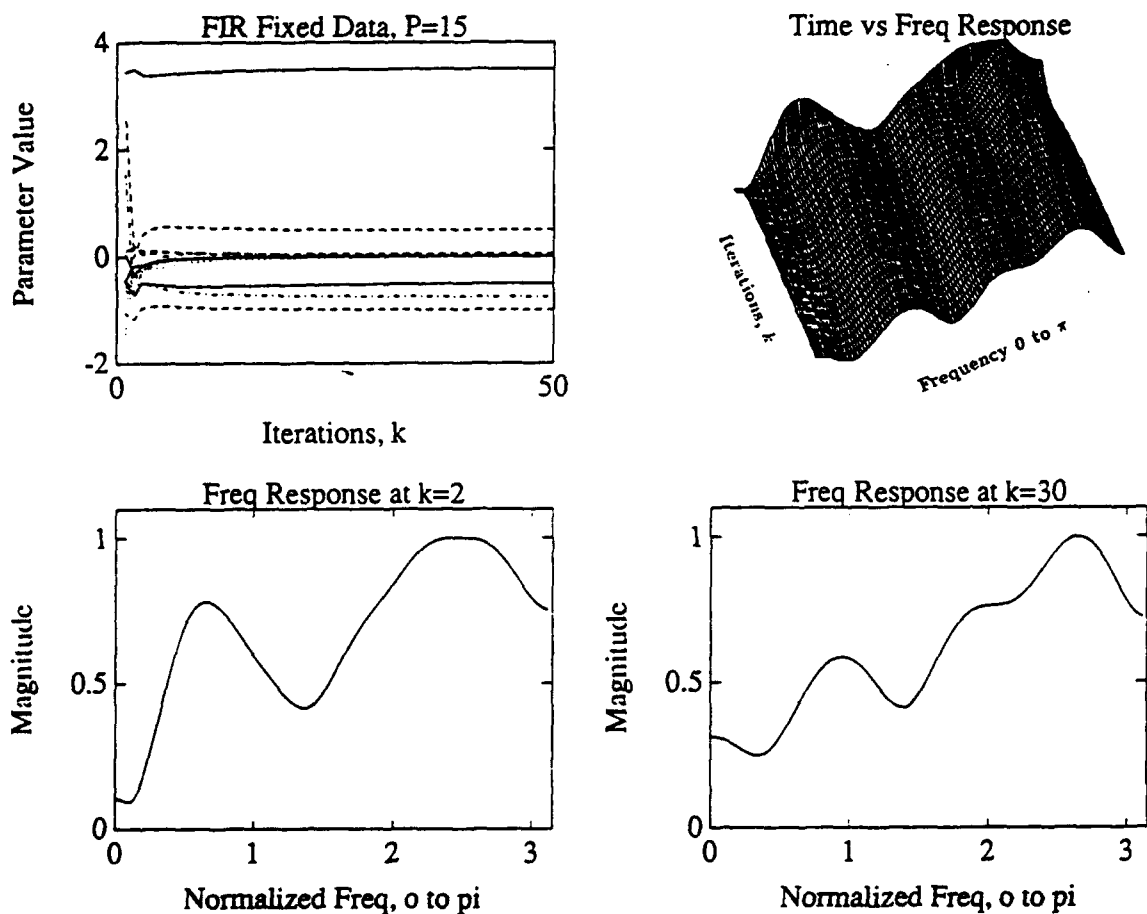


Figure A.8: Over-modeled example of the fixed data partitioned Toeplitz approximation algorithm.

The 10th-order true system was modeled with a 14th-order model. The algorithm converges rapidly, and by $k = 30$ iterations it matches the true system frequency response. It is important to note that this result was achieved with the minimum ($P = 15$) amount of data required. No noise was added to the output.

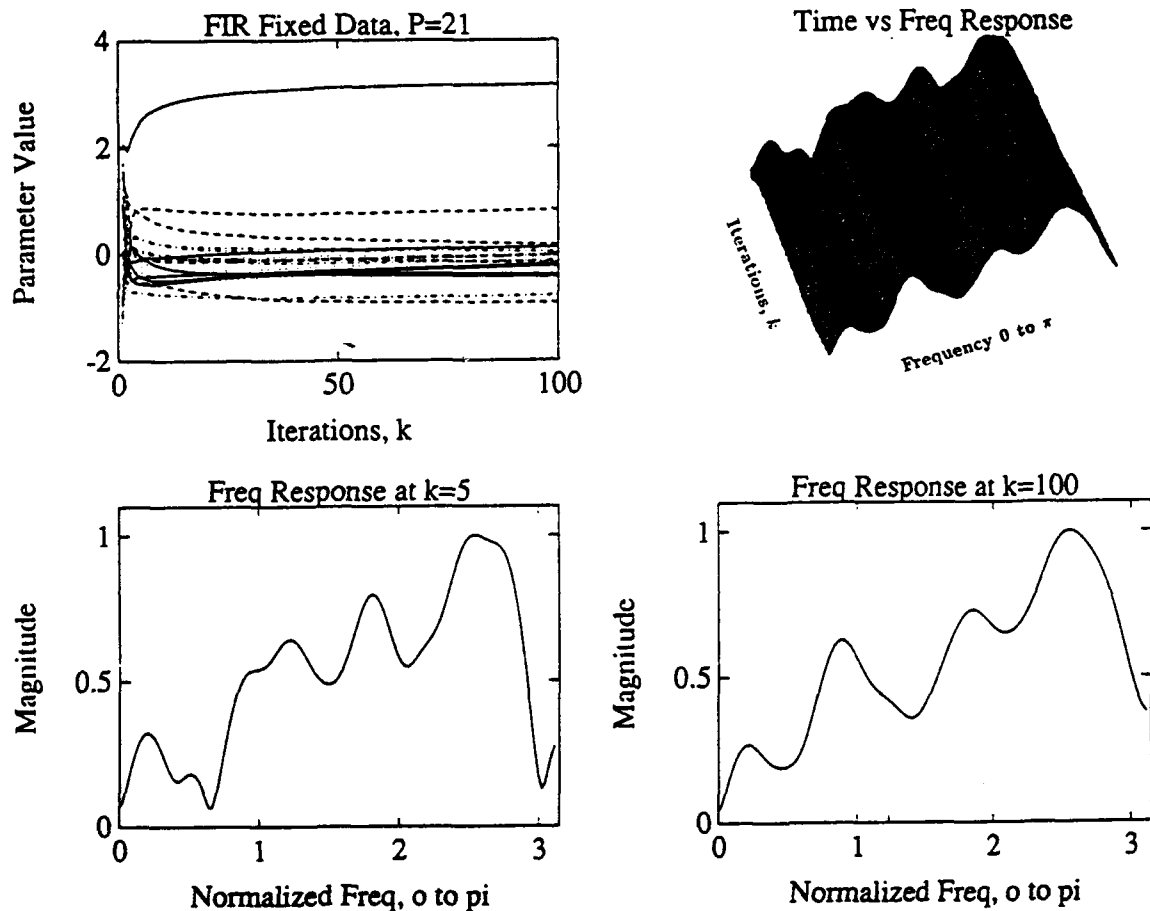


Figure A.9: Over-modeled example with 10 dB of additive noise for the fixed data partitioned Toeplitz approximation algorithm.

The 10th-order true system was modeled with a 20th-order model. The additive white noise tends to slow down the algorithm, and distorts the frequency response it is able to achieve. There was 10 dB of white noise added to the output.

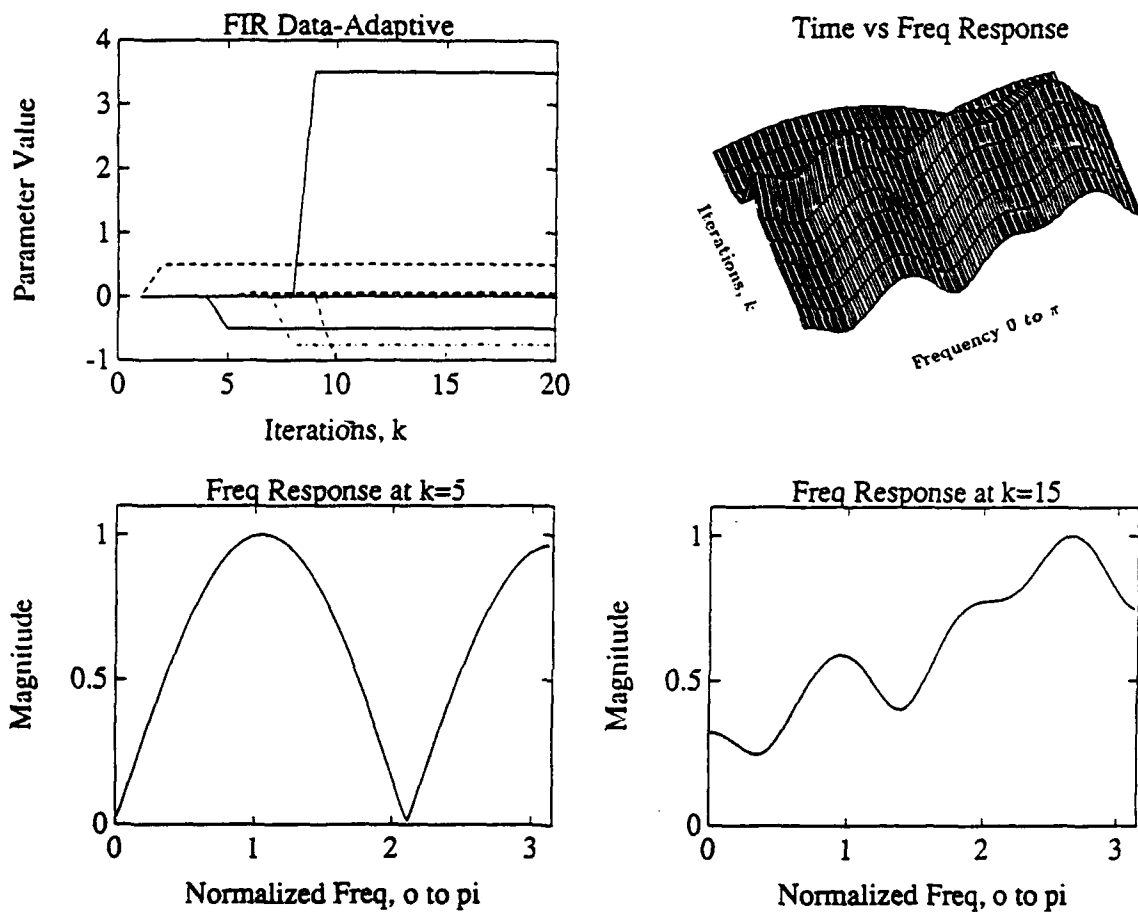


Figure A.10: Parameter tracks and frequency response of the data-adaptive Toeplitz approximation algorithm.

Observe that the algorithm converges to the true solution and frequency response in ten iterations. In this example no noise was added to the output, and the performance above is for an exact modeling of the 10th-order true system. Comparing this with the RLS results shown in Figure A.14 reveals that they behave exactly the same.

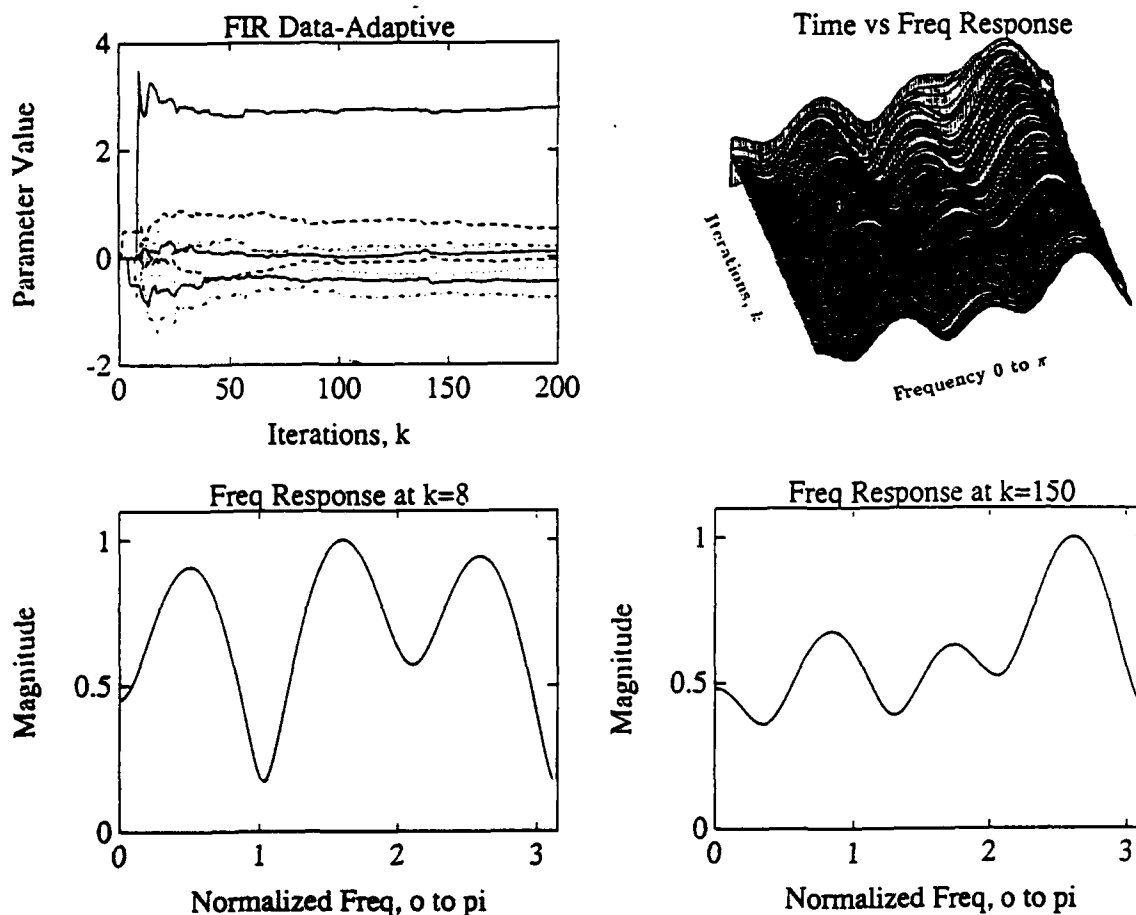


Figure A.11: Under-modeled example of the data-adaptive Toeplitz approximation algorithm.

The 10th-order true system was modeled with an 8th-order model. The algorithm tries to preserve the frequency components of the true system, however it takes a much greater number of iterations to achieve this result. No noise was added to the output.

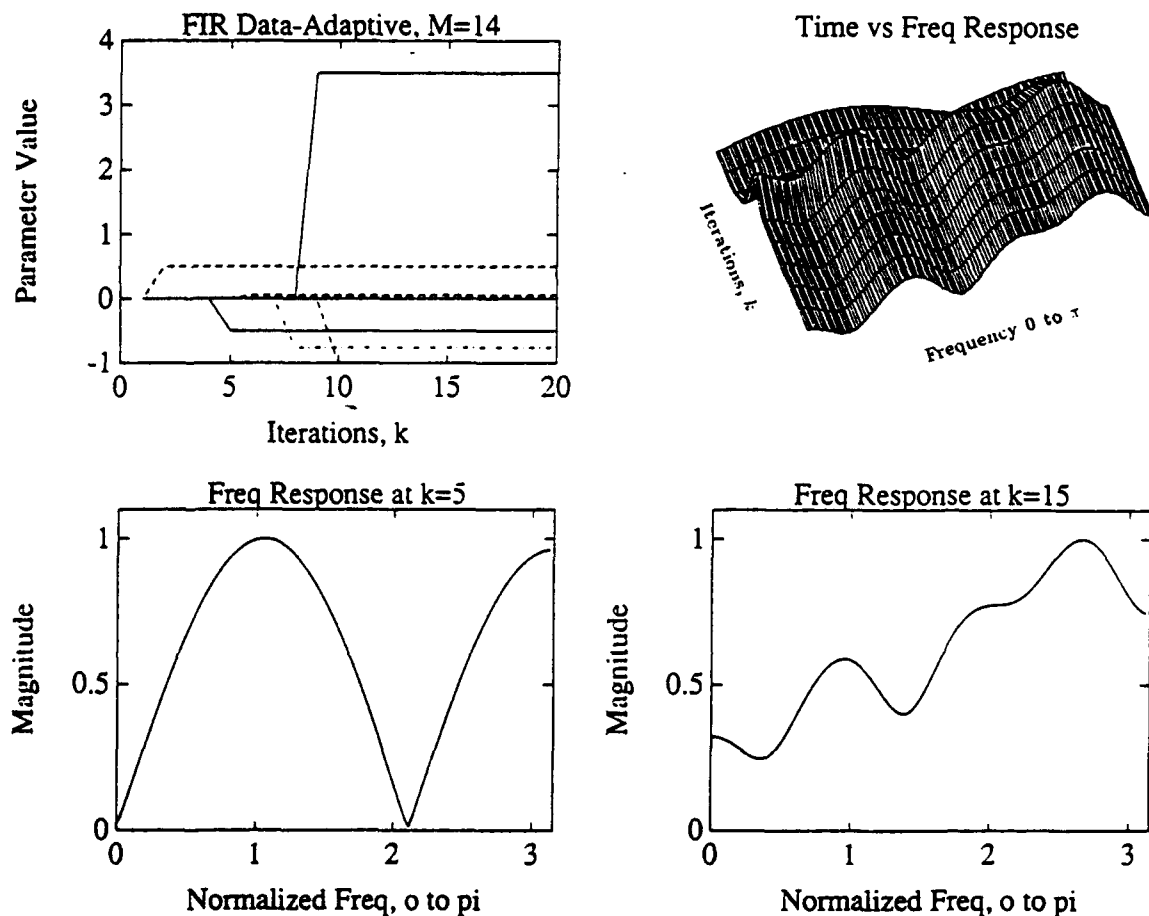


Figure A.12: Over-modeled example of the data-adaptive Toeplitz approximation algorithm.

The 10th-order true system was modeled with a 14th-order model. The algorithm converges rapidly, and for iterations $k = 10$ and higher it matches the true system frequency response. It is important to note that all of the additional parameters beyond the ten required were driven to zero by the algorithm. No noise was added to the output.

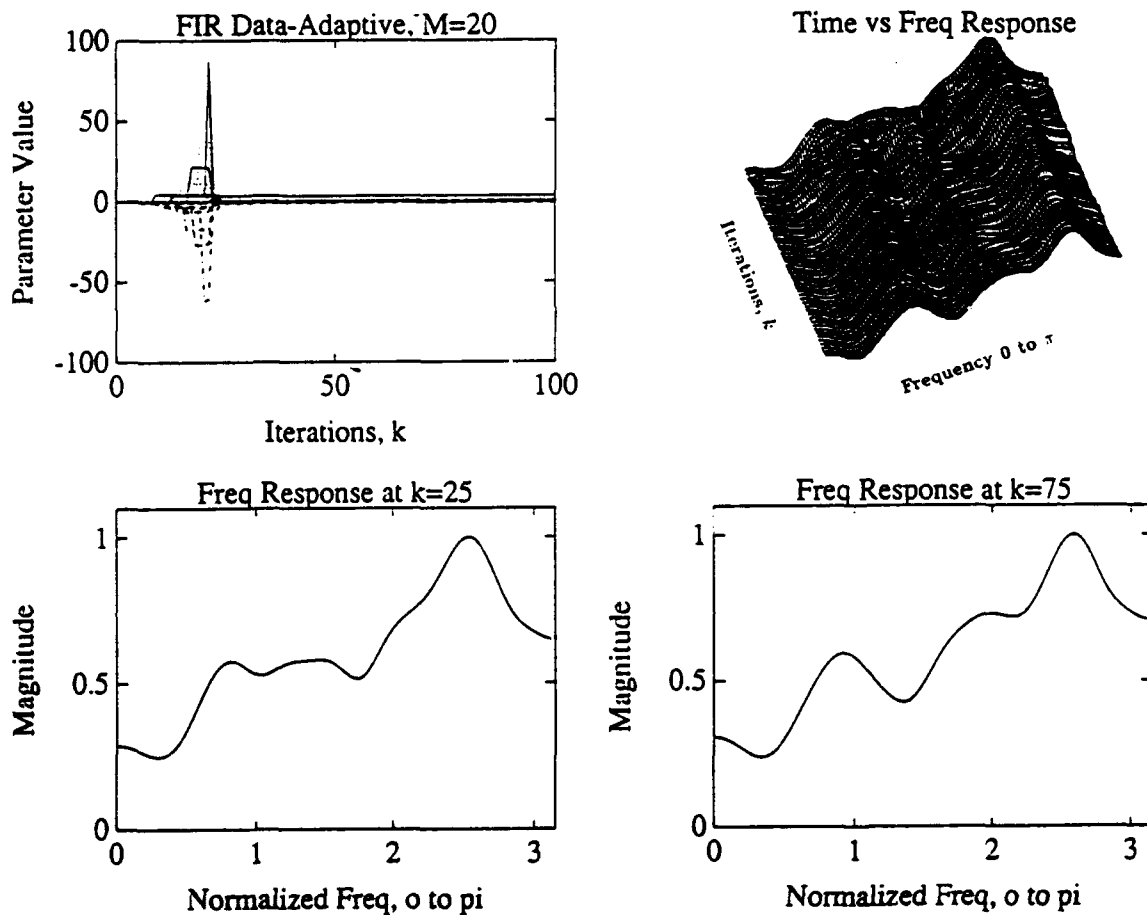


Figure A.13: Over-modeled example with 10 dB of additive noise for the data-adaptive Toeplitz approximation algorithm.

The 10th-order true system was modeled with a 20th-order model. The additive white noise tends to slow down the algorithm, but only slightly distorts the frequency response it is able to achieve. There was 10 dB of white noise added to the output.

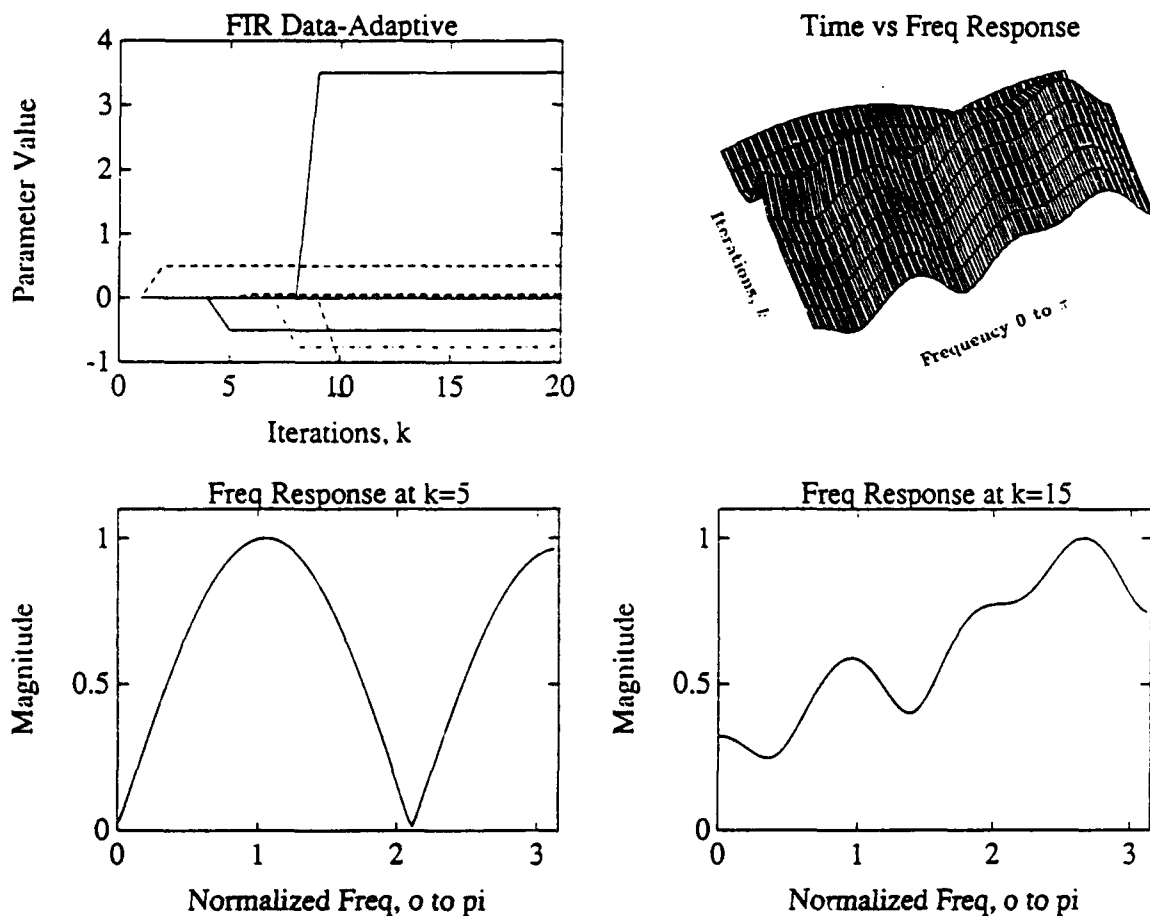


Figure A.14: Parameter tracks and frequency response of the FIR RLS algorithm.

Observe that the algorithm converges to the true solution and frequency response in ten iterations. In this example no noise was added to the output, and the performance above is for an exact modeling of the 10th-order true system. Comparing this with the data-adaptive Toeplitz approximation algorithm results shown in Figure A.10 reveals that they behave exactly the same.

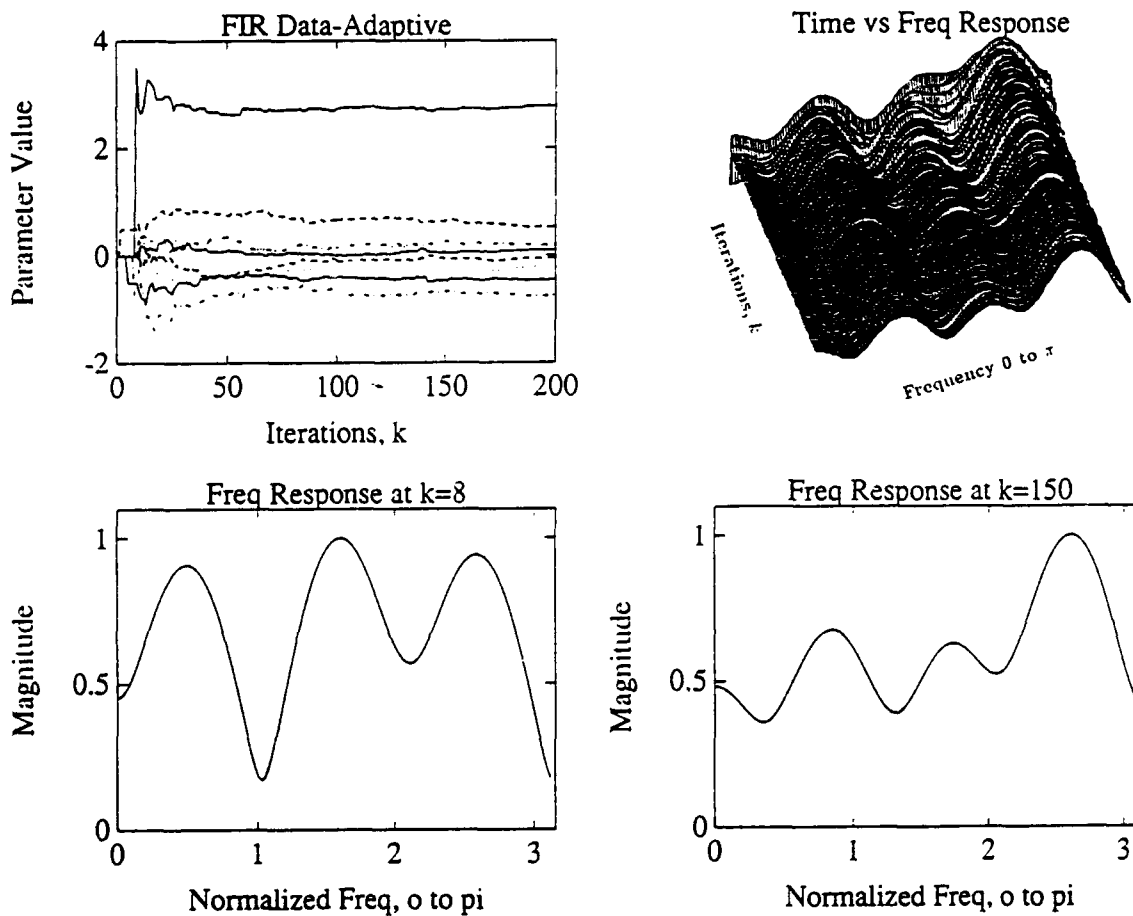


Figure A.15: Under-modeled example of the FIR RLS algorithm. The 10th-order true system was modeled with an 8th-order model. The algorithm tries to preserve the frequency components of the true system, however it takes a much greater number of iterations to achieve this result. No noise was added to the output.

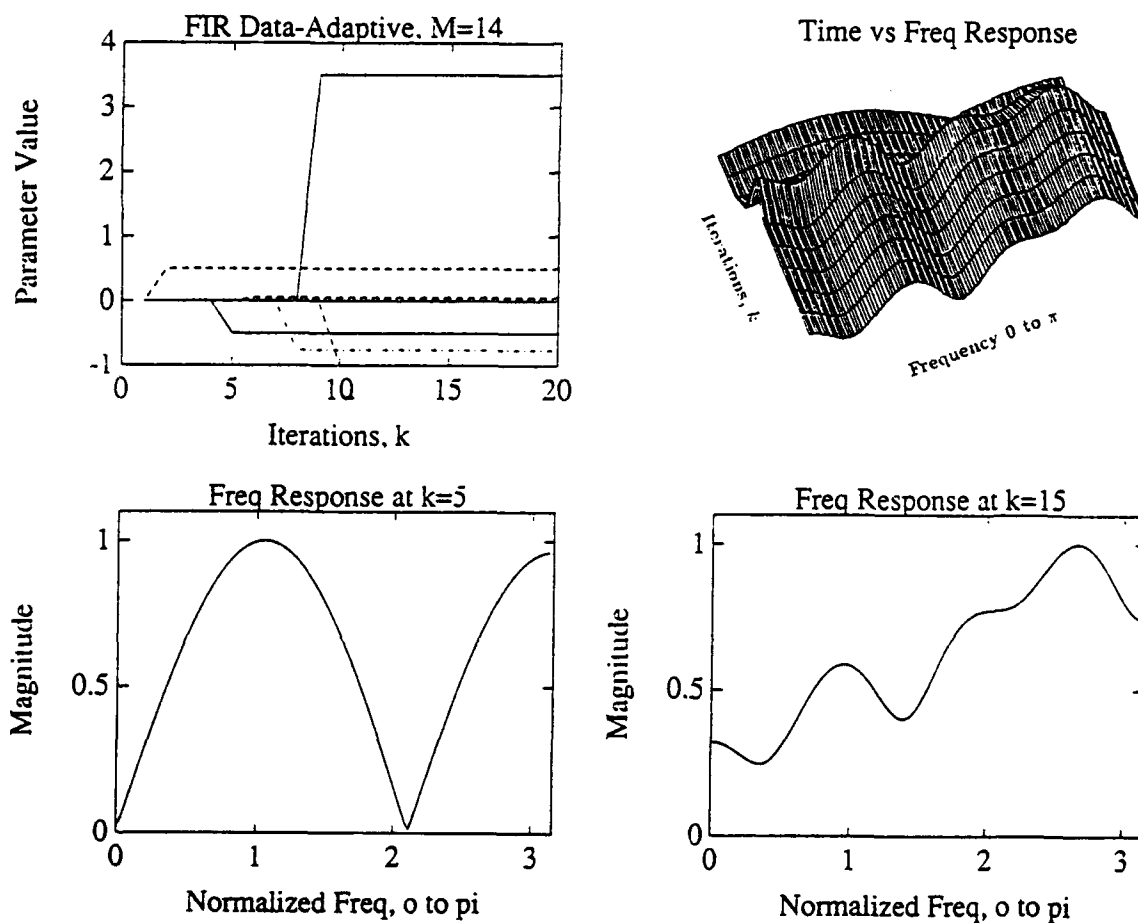


Figure A.16: Over-modeled example of the FIR RLS algorithm. The 10th-order true system was modeled with a 14th-order model. The algorithm converges rapidly, and by iteration $k = 10$ and higher it matches the true system frequency response. It is important to note that all of the additional parameters beyond the ten required were driven to zero by the algorithm. No noise was added to the output.

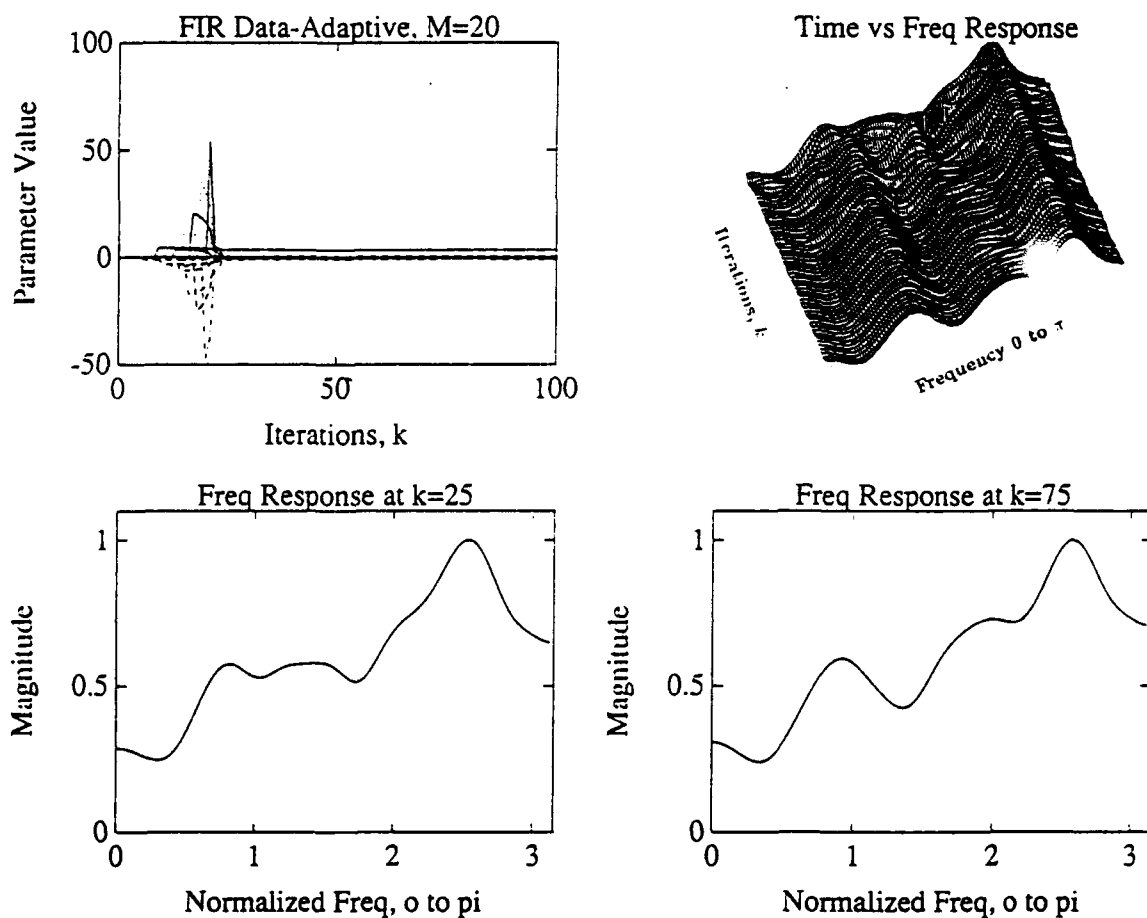


Figure A.17: Over-modeled example with 10 dB of additive noise for the data-adaptive Toeplitz approximation algorithm.

The 10th-order true system was modeled with a 20th-order model. The additive white noise tends to slow down the algorithm, but only slightly distorts the frequency response it is able to achieve. There was 10 dB of white noise added to the output.

APPENDIX B: FIXED DATA IIR SYSTEM SIMULATIONS

This appendix contains plots corresponding to the fixed data IIR algorithms developed in Chapter IV. Figure B.1 shows the performance of the unmodified Toeplitz approximation algorithm (4.8). This plot also serves as a reference of the true system (4.9) frequency response. The remainder of the plots, Figures B.2–B.9 illustrate the convergence performance of the fixed data Toeplitz approximation with diagonal perturbation algorithm followed by the fixed data partitioned Toeplitz approximation algorithm.

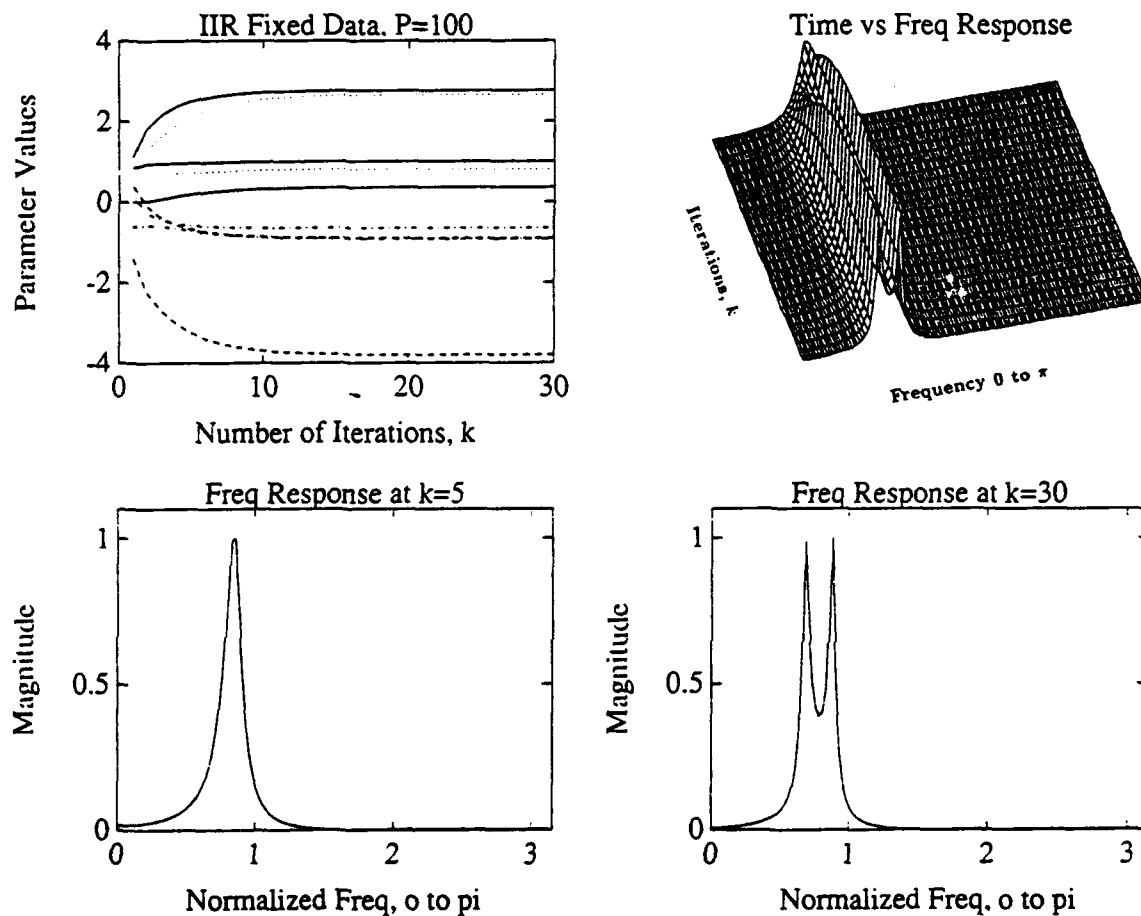


Figure B.1: The basic fixed data IIR Toeplitz approximation algorithm. This plot shows the parameter tracks and frequency response of the basic Toeplitz approximation algorithm (4.8). The correlation matrix was formed with $P = 100$ data points, and there was no additive noise in the output. The algorithm converges rapidly under these conditions, as can be seen by the flat parameter tracks. Also notice that the frequency response changes little as the number of iterations increase. The frequency response plot for $k = 30$ iterations represents the frequency response of the true system.

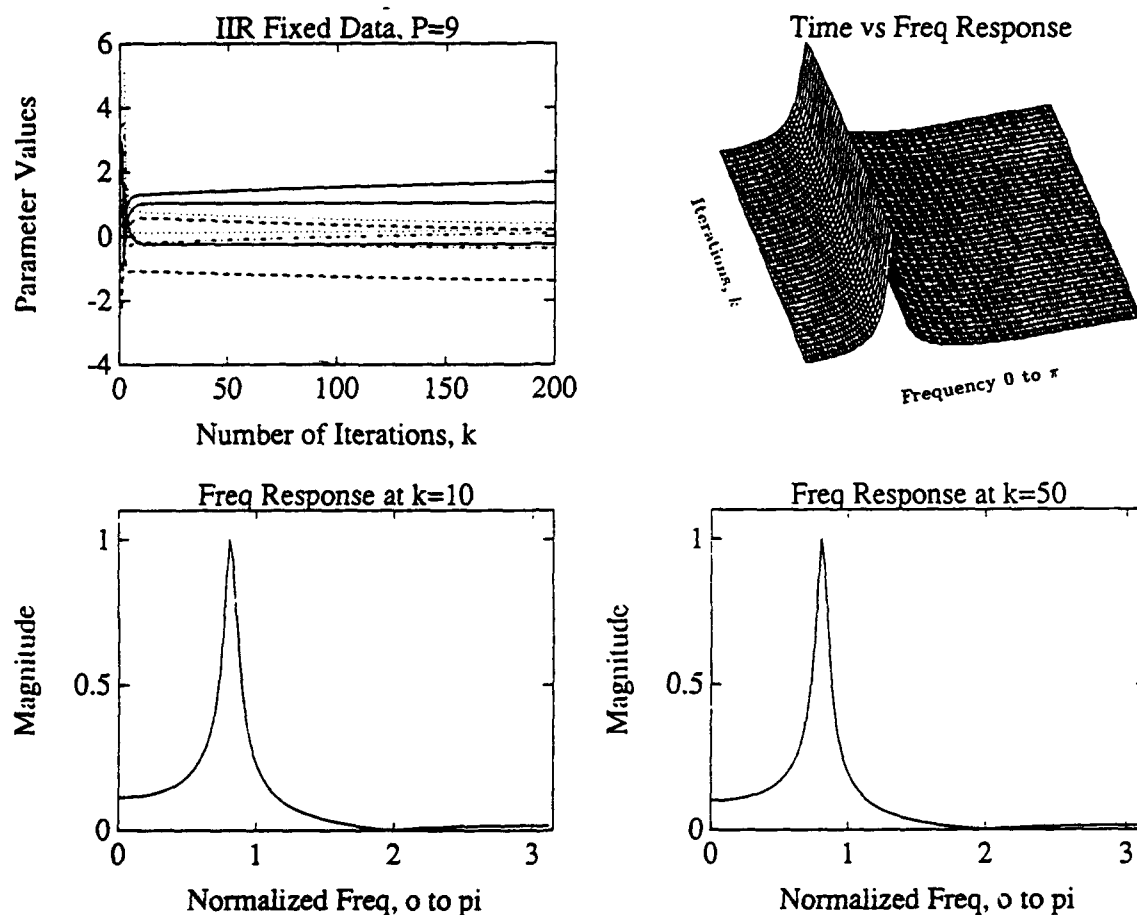


Figure B.2: Parameter tracks and frequency response of the fixed data Toeplitz approximation with diagonal perturbation algorithm.

Observe that when the minimum number of data points ($P = 9$) were used to form the correlation matrix, the algorithm converges very slowly. The frequency response plot for $k = 50$ iterations is not very close to the response of the true system shown in Figure B.1. No noise was added to the output, the perturbation parameters were $\sigma_x = 1$ and $\sigma_y = 10$, and the true system was exactly modeled with a 4th-order model.

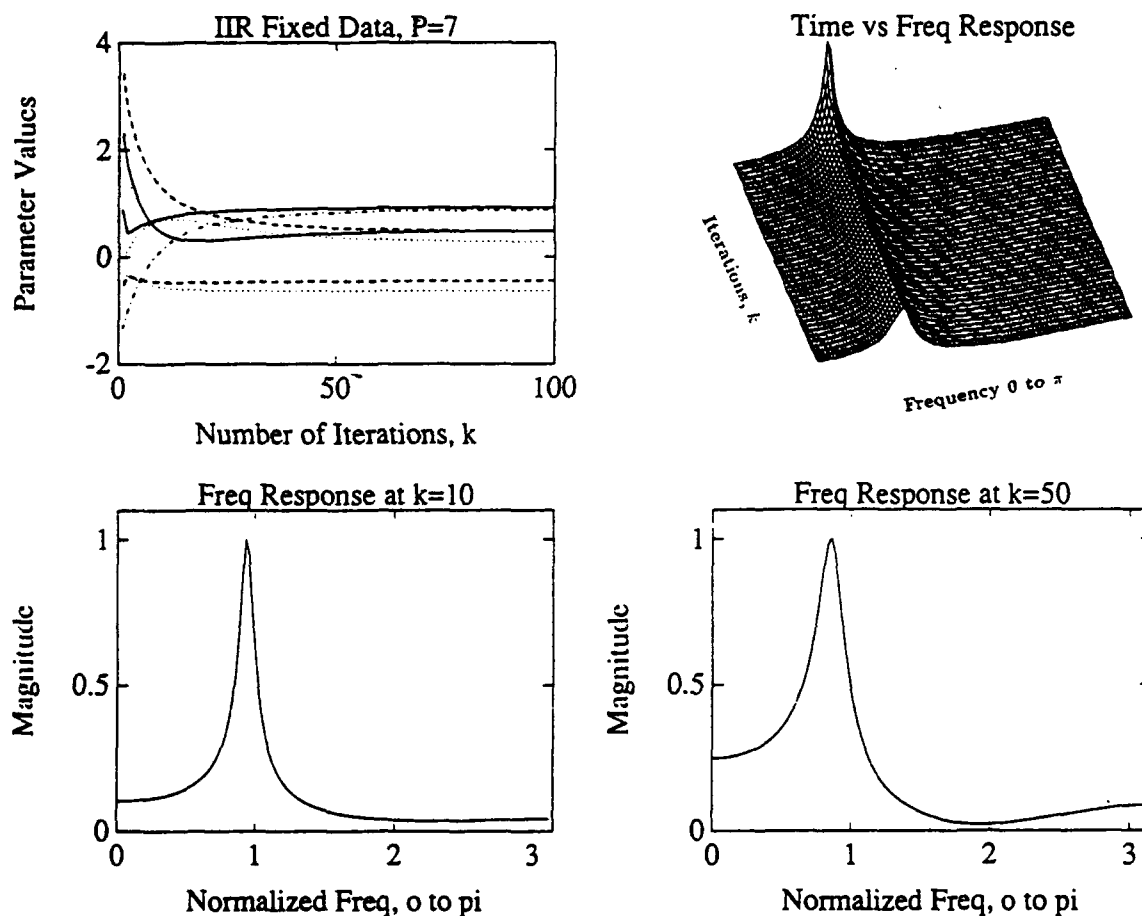


Figure B.3: Under-modeled example of the fixed data Toeplitz approximation with diagonal perturbation algorithm.

The 4th-order true system was modeled with an 3rd-order model. The algorithm does a poor job preserving the frequency components of the true system, and it takes a much greater number of iterations to achieve this result. No noise was added to the output, and the perturbation parameters were $\sigma_x = 4$ and $\sigma_y = 50$.

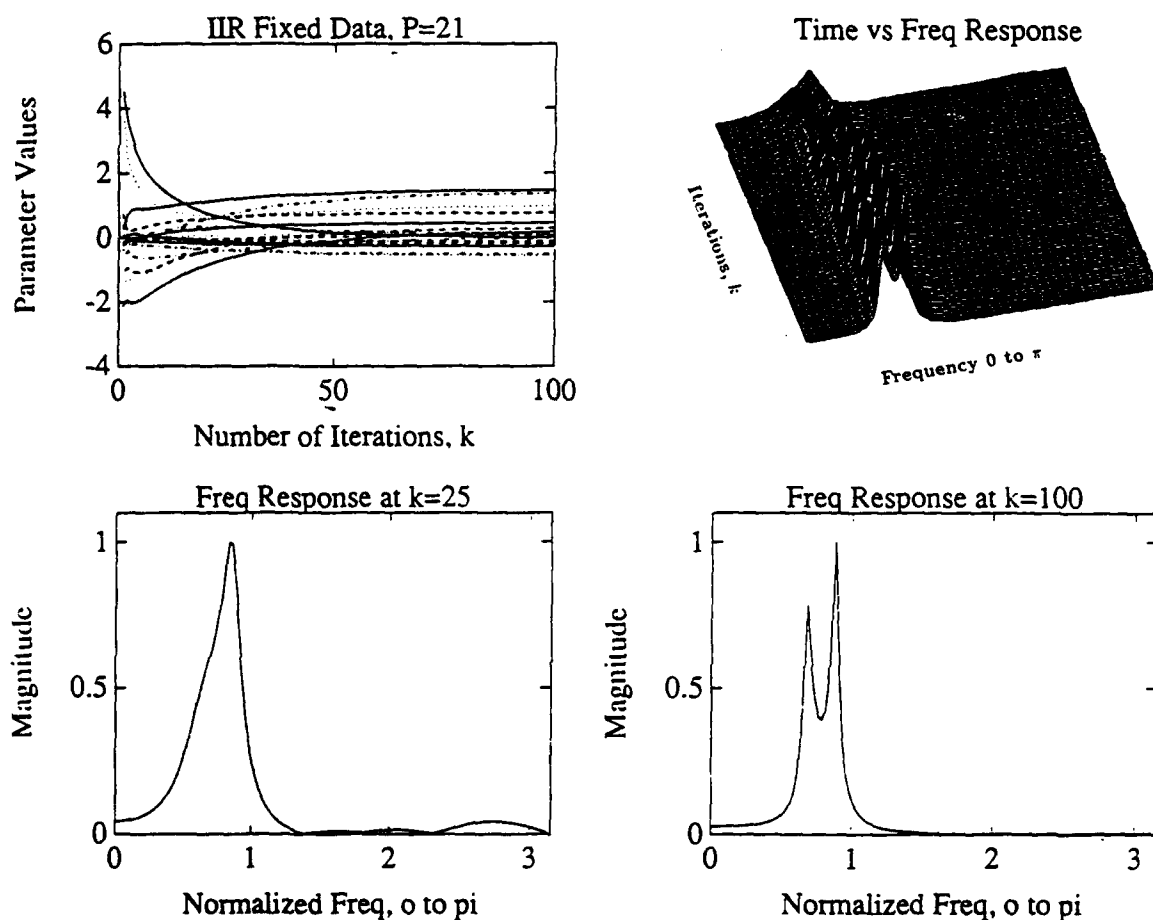


Figure B.4: Over-modeled example of the fixed data Toeplitz approximation with diagonal perturbation algorithm.

The 4th-order true system was modeled with a 10th-order model. The algorithm converges more rapidly than the previous two examples, and by $k = 100$ iterations it nearly matches the true system frequency response. It is important to note that this result was achieved with the minimum ($P = 21$) amount of data required. No noise was added to the output, and the perturbation parameters were $\sigma_x = 3$ and $\sigma_y = 100$.

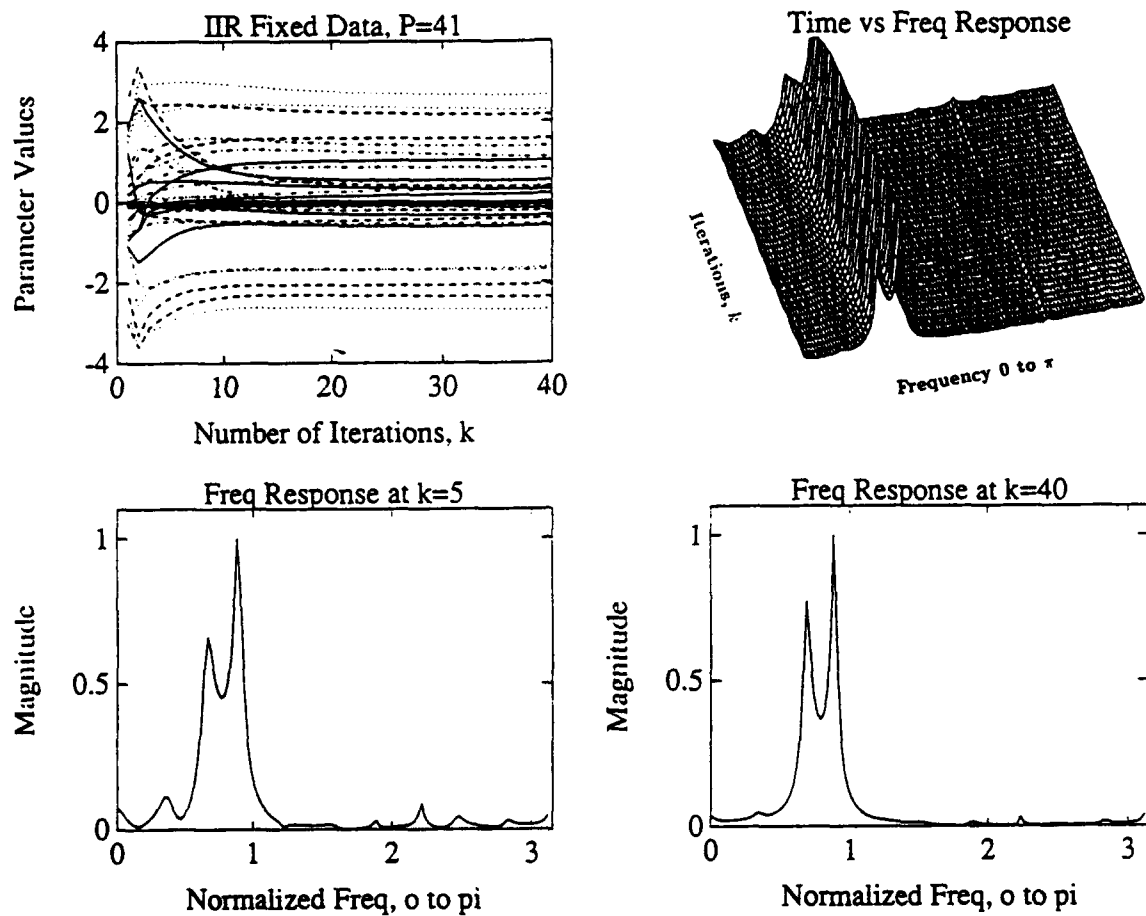


Figure B.5: Over-modeled example with 10 dB of additive noise for the fixed data Toeplitz approximation with diagonal perturbation algorithm. The 4th-order true system was modeled with a 20th-order model. The additive white noise tends to slow down the algorithm, but only slightly distorts the frequency response it is able to achieve. The perturbation parameters were $\sigma_x = 5$ and $\sigma_y = 100$, and 10 dB of white noise was added to the output.

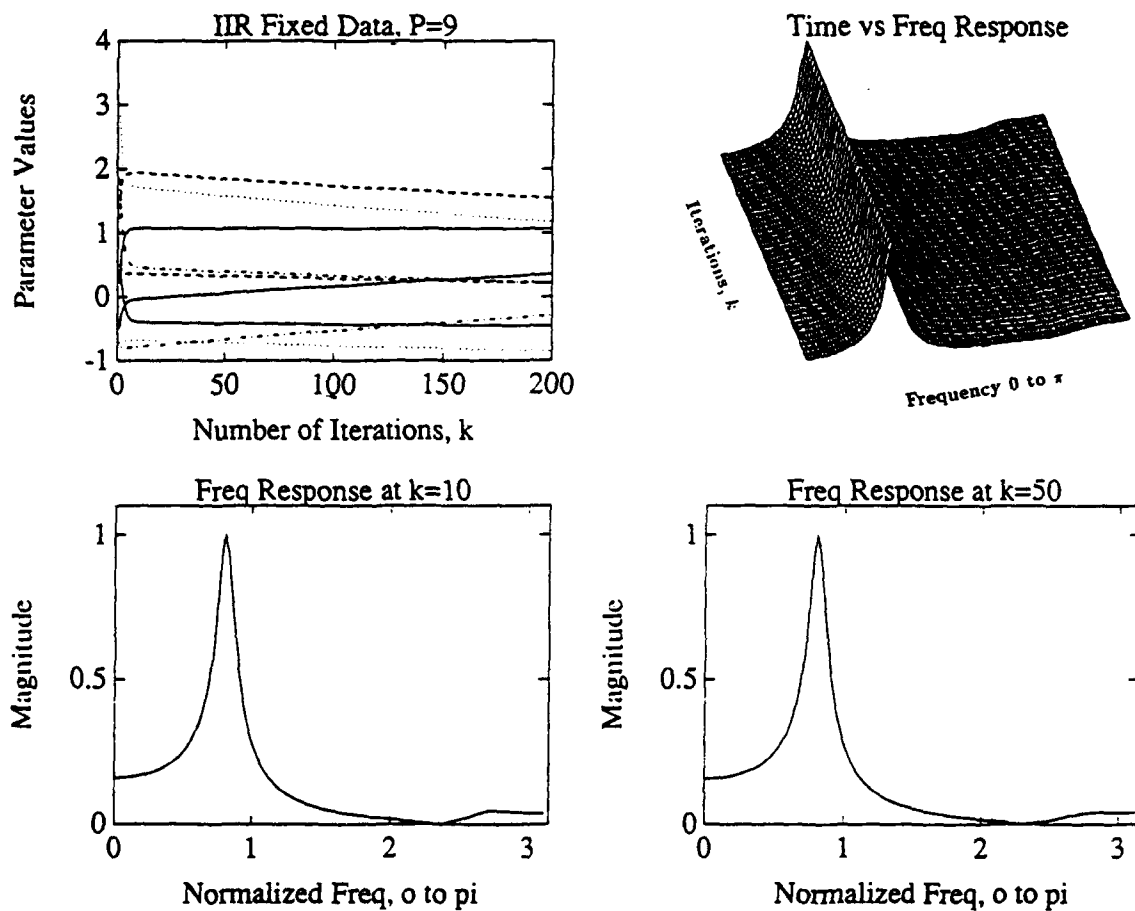


Figure B.6: Parameter tracks and frequency response of the fixed data partitioned Toeplitz approximation algorithm.

Observe that when the minimum number of data points ($P = 9$) were used to form the correlation matrix, the algorithm converges very quickly. The frequency response plot for $k = 50$ iterations is not very close to the response of the true system shown in Figure B.1. No noise was added to the output, and the true system was exactly modeled with a 4th-order model.

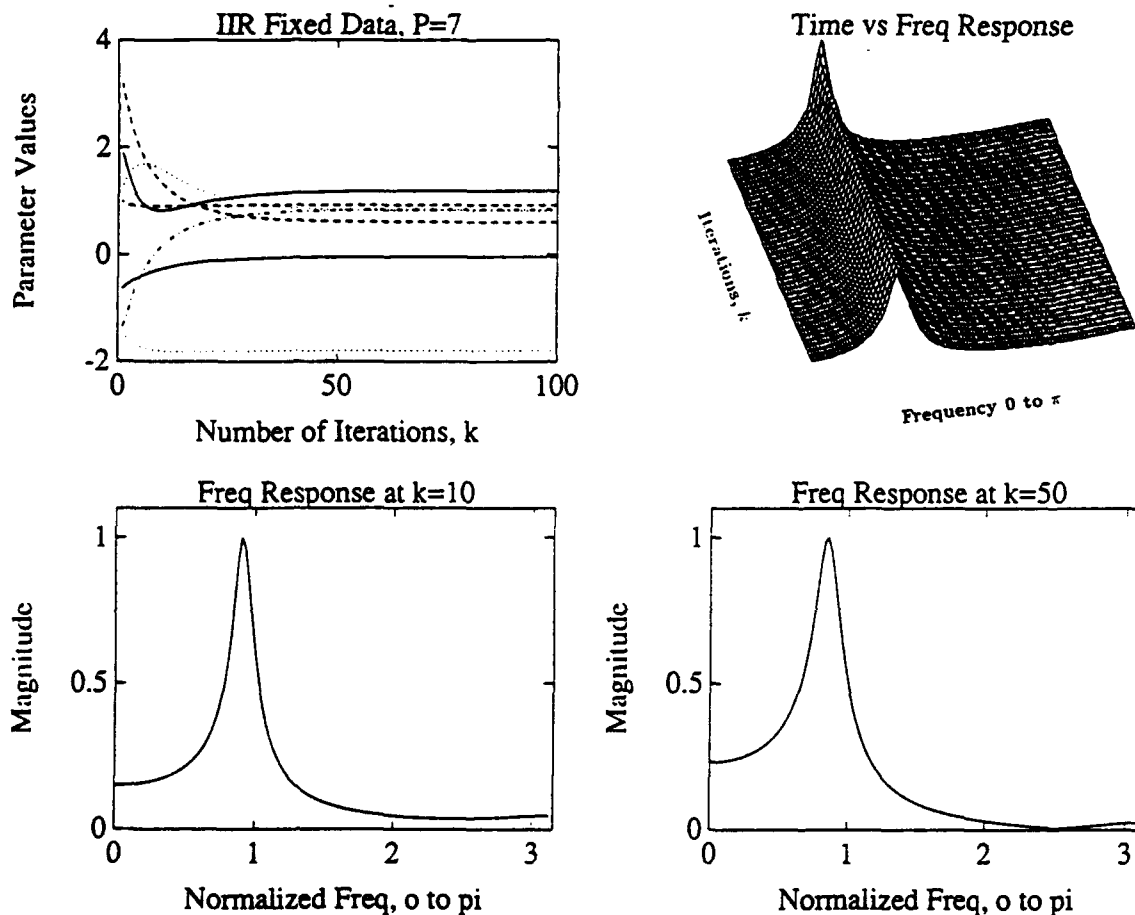


Figure B.7: Under-modeled example of the fixed data partitioned Toeplitz approximation algorithm.

The 4th-order true system was modeled with an 3rd-order model. The algorithm does a poor job preserving the frequency components of the true system, and it takes a much greater number of iterations to achieve this result. No noise was added to the output.

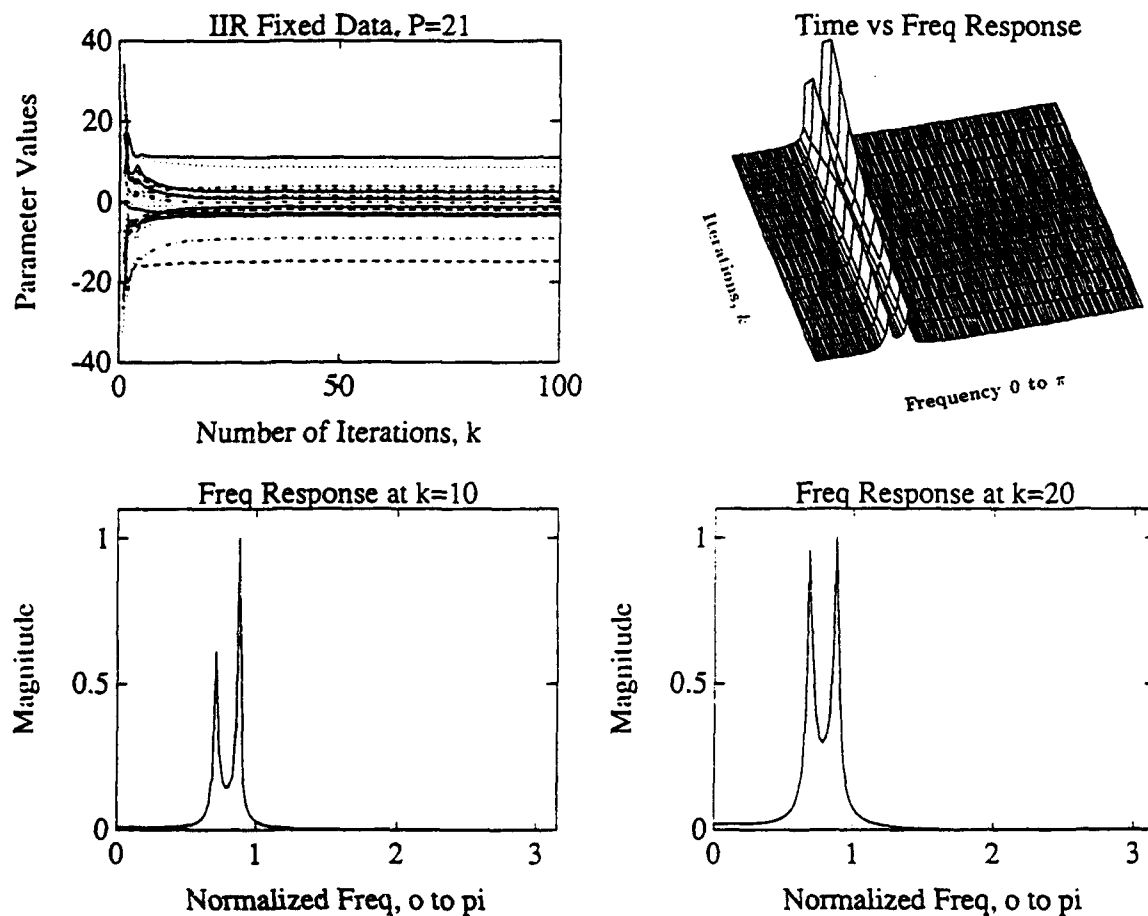


Figure B.8: Over-modeled example of the fixed data partitioned Toeplitz approximation algorithm.

The 4th-order true system was modeled with a 10th-order model. The algorithm converges rapidly, and by $k = 20$ iterations it matches the true system frequency response. It is important to note that this result was achieved with the minimum ($P = 21$) amount of data required. No noise was added to the output.

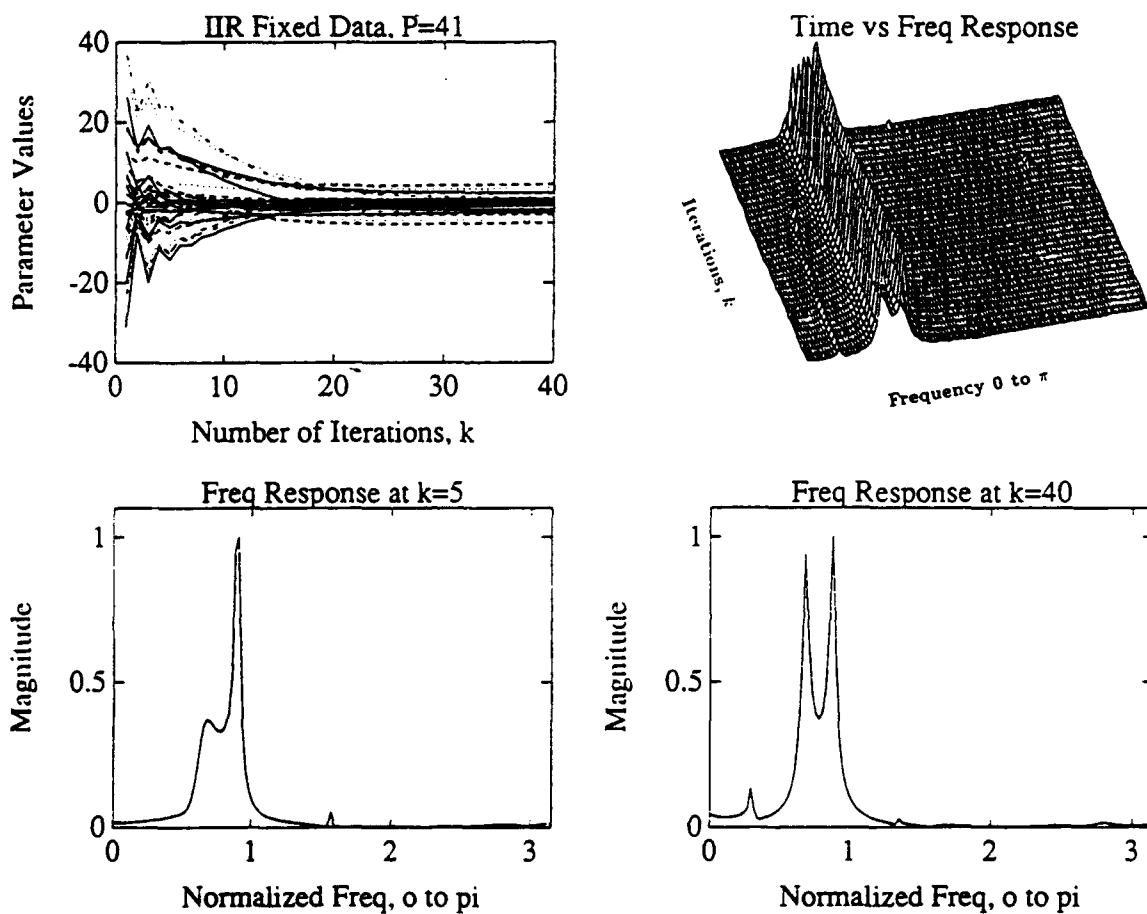


Figure B.9: Over-modeled example with 10 dB of additive noise for the fixed data partitioned Toeplitz approximation algorithm.

The 4th-order true system was modeled with a 20th-order model. The additive white noise tends to slow down the algorithm, but only slightly distorts the frequency response it is able to achieve. There was 10 dB of white noise added to the output.

APPENDIX C: DATA-ADAPTIVE IIR SYSTEM SIMULATIONS

This appendix contains plots corresponding to the data-adaptive IIR algorithms developed in Chapter IV. Figure C.1 shows the performance of the unmodified *fixed data* Toeplitz approximation algorithm (4.8). This plot also serves as a reference of the true system (4.9) frequency response. The remainder of the plots, Figures C.2–C.4, illustrate the performance of the data-adaptive Toeplitz approximation algorithm followed by the IIR RLS algorithm.

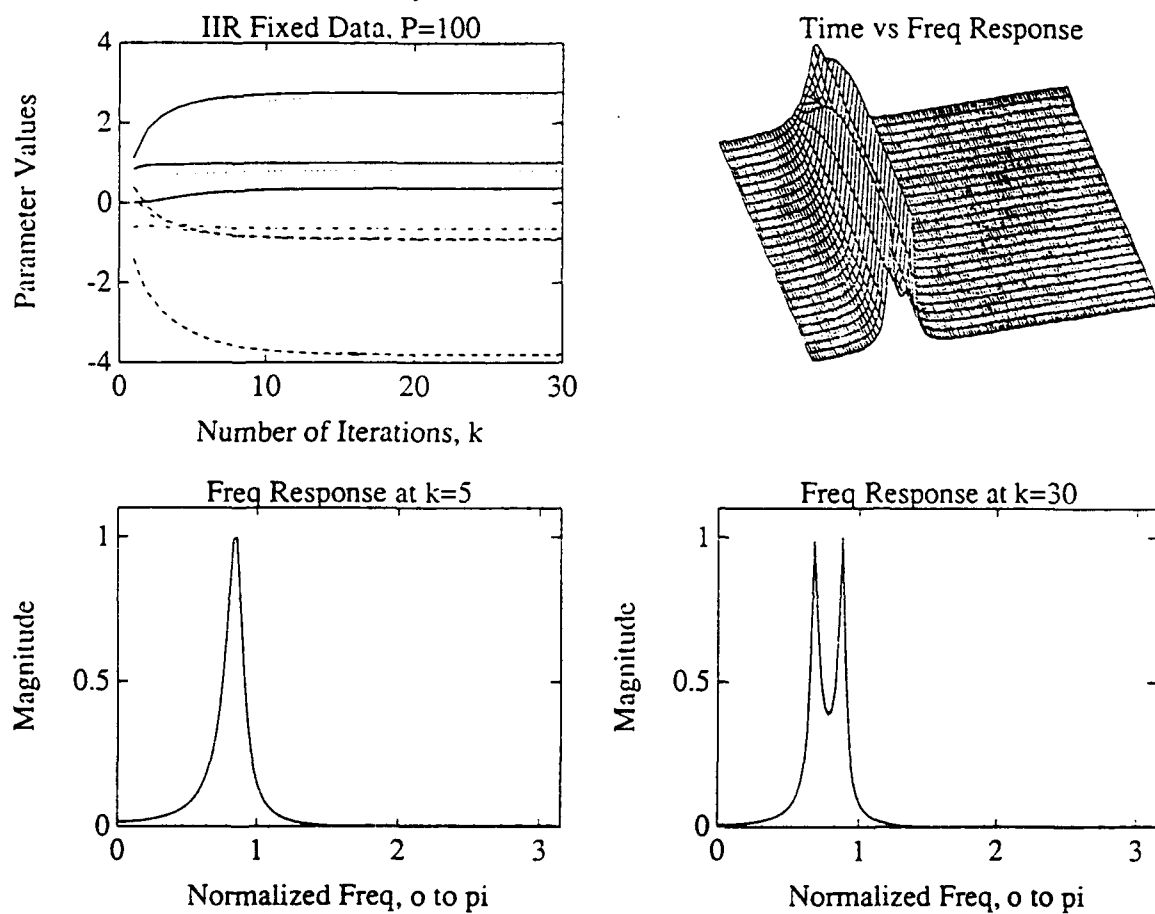


Figure C.1: The basic fixed data IIR Toeplitz approximation algorithm. This plot shows the parameter tracks and frequency response of the basic Toeplitz approximation algorithm (4.8). The correlation matrix was formed with $P = 100$ data points, and there was no additive noise in the output. The algorithm converges rapidly under these conditions, as can be seen by the flat parameter tracks. Also notice that the frequency response changes little as the number of iterations increase. The frequency response plot for $k = 30$ iterations represents the frequency response of the true system.

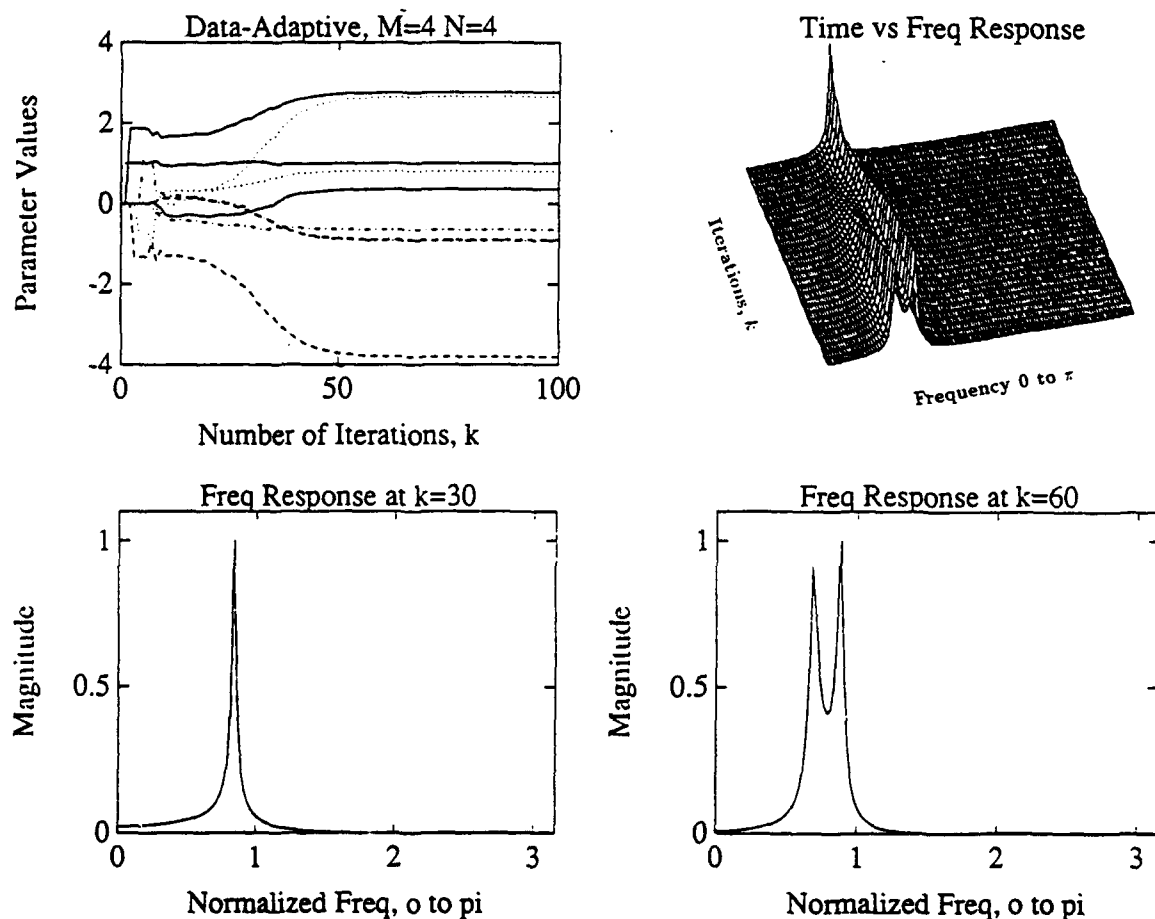


Figure C.2: Parameter tracks and frequency response of the data-adaptive Toeplitz approximation algorithm.

Observe that the algorithm converges to the true solution and frequency response in about 60 iterations. In this example no noise was added to the output, and the performance above is for an exact modeling of the 4th-order true system. Comparing this with the RLS results shown in Figure B.18 reveals that they do not behave the same as in the FIR case.

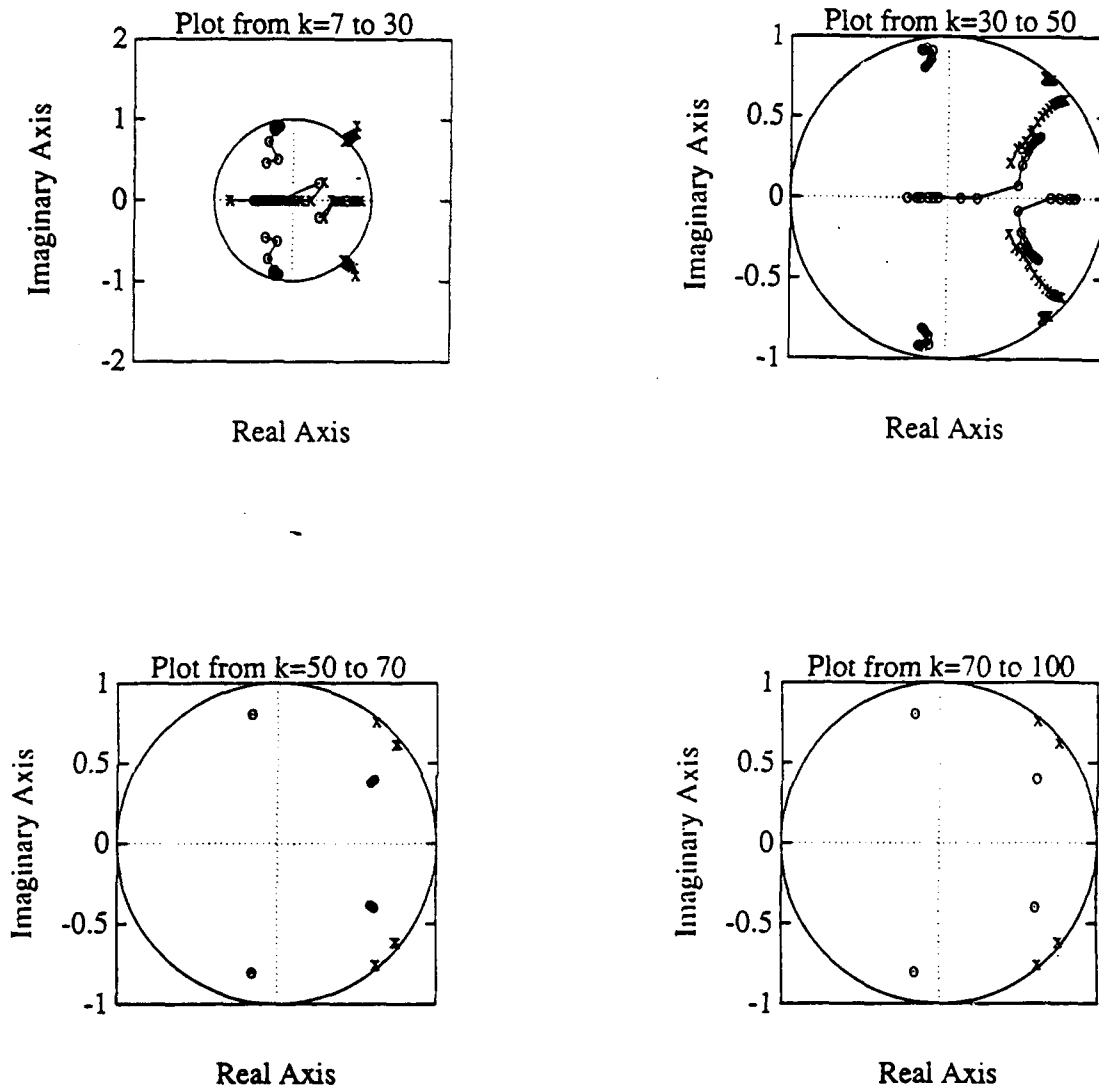


Figure C.3: Pole-zero plots for the data-adaptive Toeplitz approximation algorithm.

Observe that as the algorithm converges to the true solution, the poles (x's) move inside the unit circle. After about 50 iterations they stop moving, and their location corresponds to the frequency response shown in Figure B.10. Finally, we see that there is almost no movement as the algorithm iterates from $k = 70$ to 100 iterations.

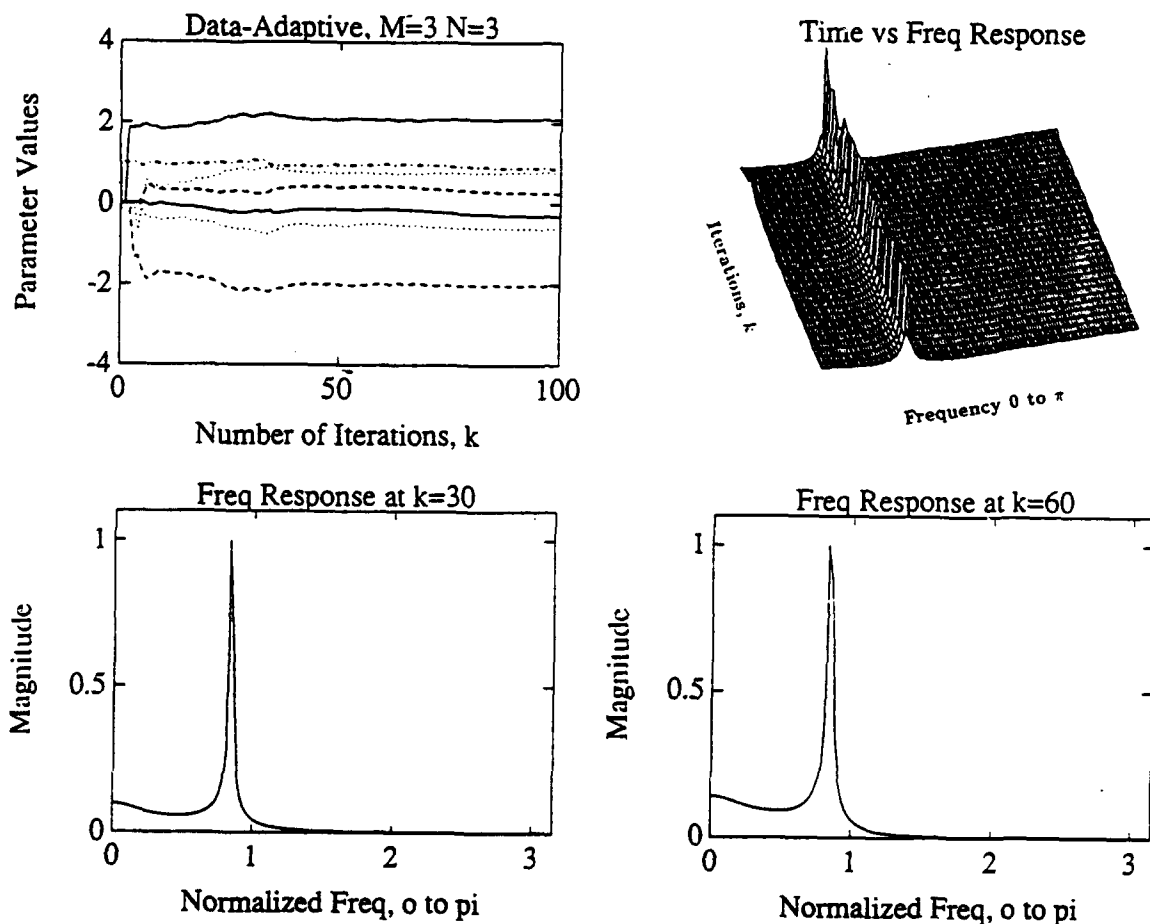


Figure C.4: Under-modeled example of the data-adaptive Toeplitz approximation algorithm.

The 4th-order true system was modeled with a 3rd-order model. The algorithm tries to preserve the frequency components of the true system, but with only three poles and zeros it is unable to preserve both of the main components. No noise was added to the output.

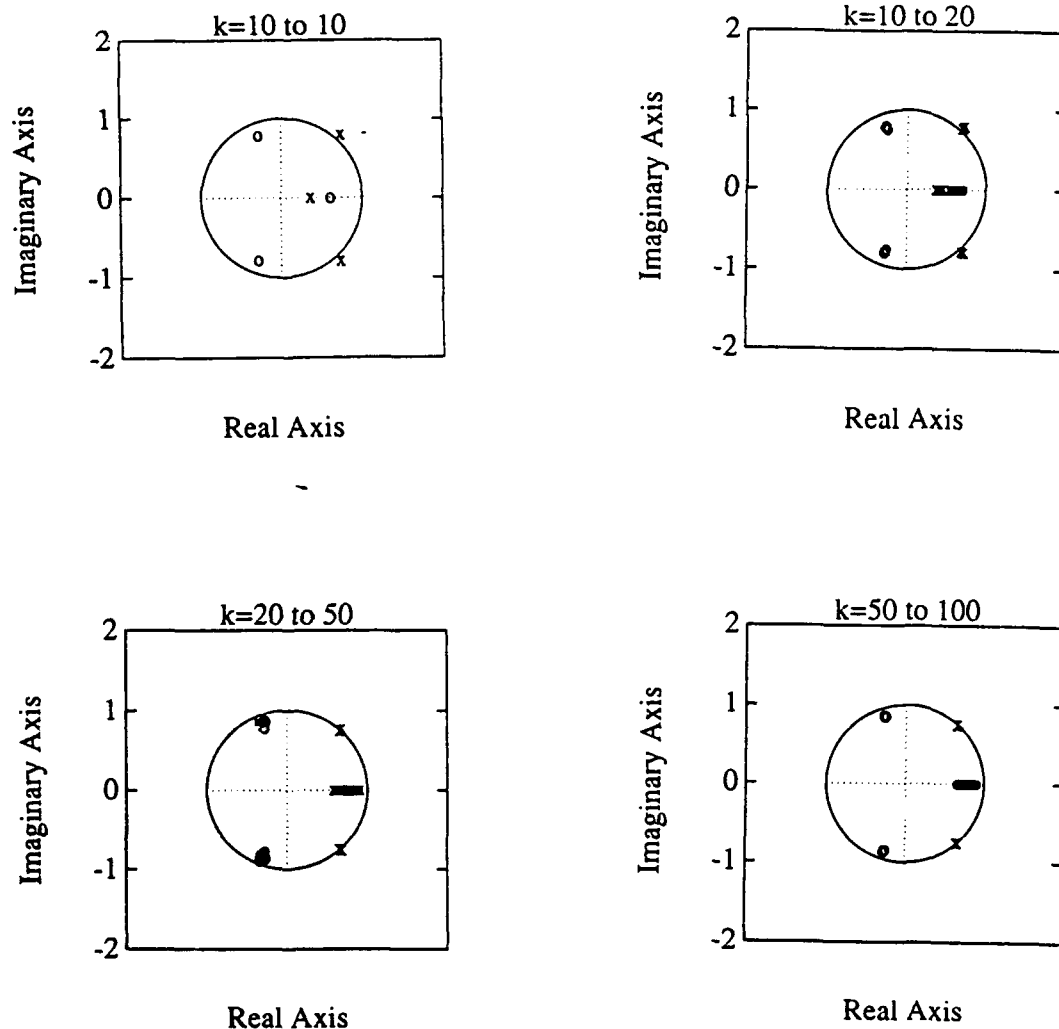


Figure C.5: Pole-zero plots for the under-modeled data-adaptive Toeplitz approximation algorithm.

Observe that as the algorithm converges, the poles (x's) still move inside the unit circle. After about 50 iterations their location corresponds to the frequency response shown in Figure B.12. Finally, we see that the pole-zero pair on the real axis tend to cancel each other, which explains why there is only one primary frequency component in Figure B.12.

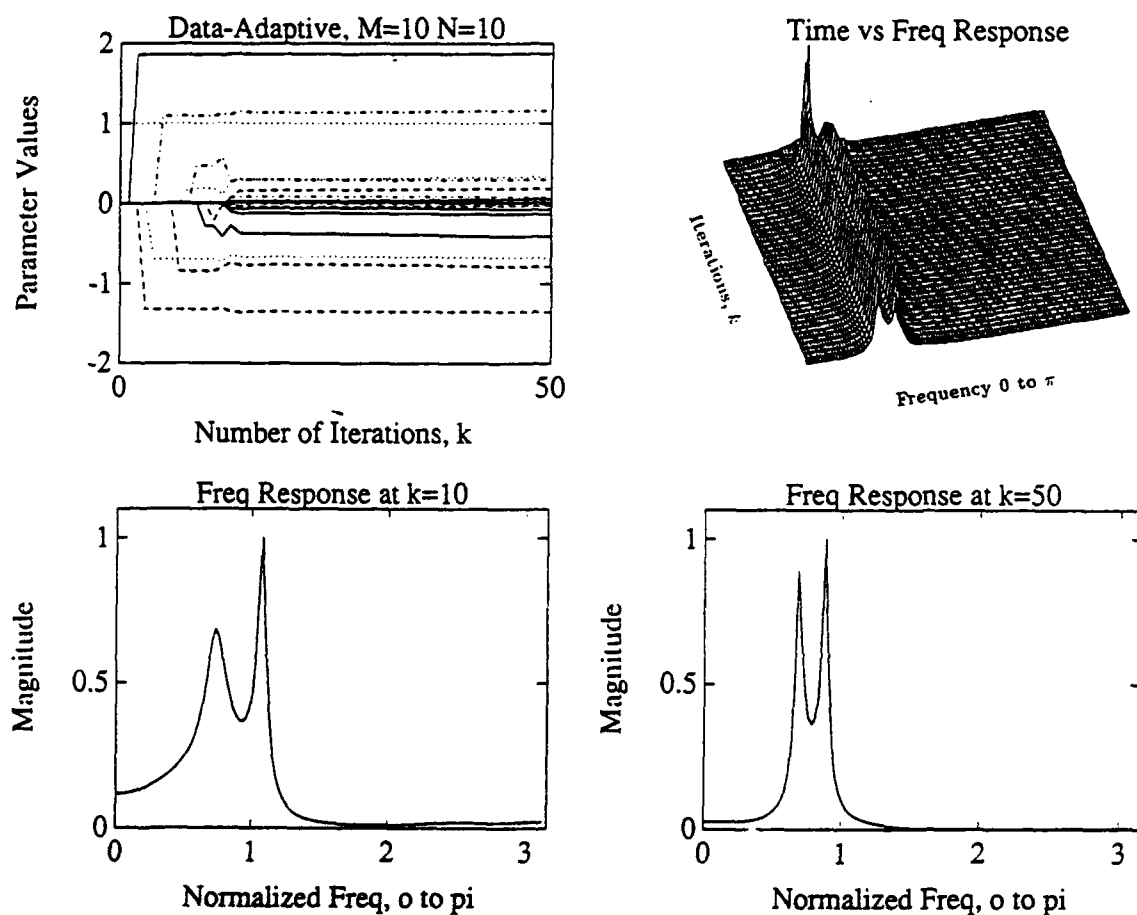


Figure C.6: Over-modeled example of the data-adaptive Toeplitz approximation algorithm.

The 4th-order true system was modeled with a 10th-order model. The algorithm converges rapidly, and by $k = 50$ iterations it matches the true system frequency response. It is important to note that all of the additional pole zero pairs beyond the four required were driven to cancel each other by the algorithm. No noise was added to the output.

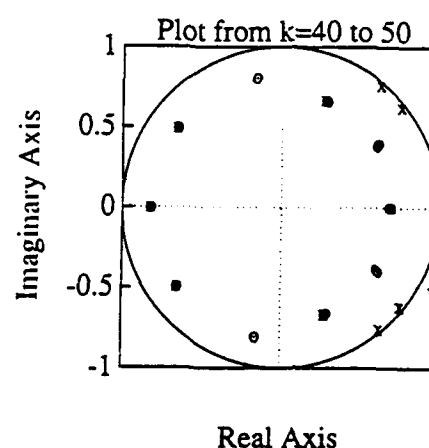
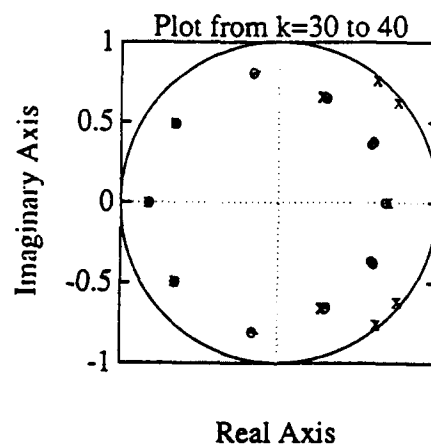
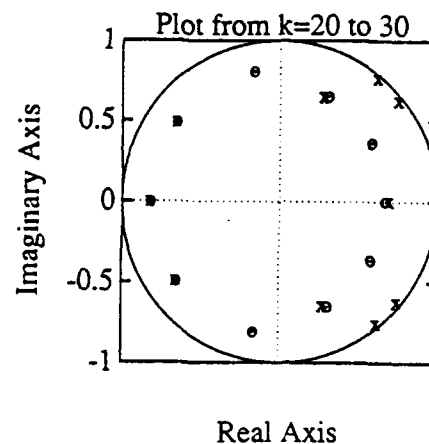
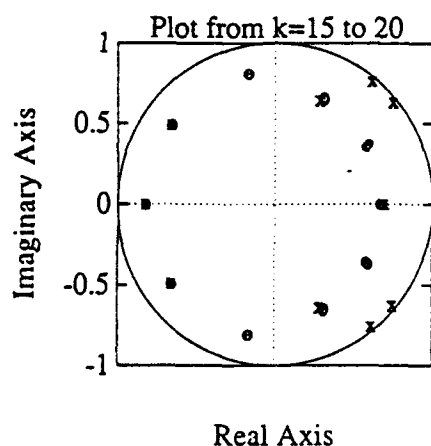


Figure C.7: Pole-zero plots for the over-modeled data-adaptive Toeplitz approximation algorithm.

Observe that as the algorithm converges, the poles (x's) are all inside the unit circle. After about 50 iterations their location corresponds to the frequency response shown in Figure B.14. Finally, we see that the pole-zero pairs different from those shown in Figure B.11 are overlapping and tend to cancel each other, which explains why both of the dominant frequency components in Figure B.10 are present.

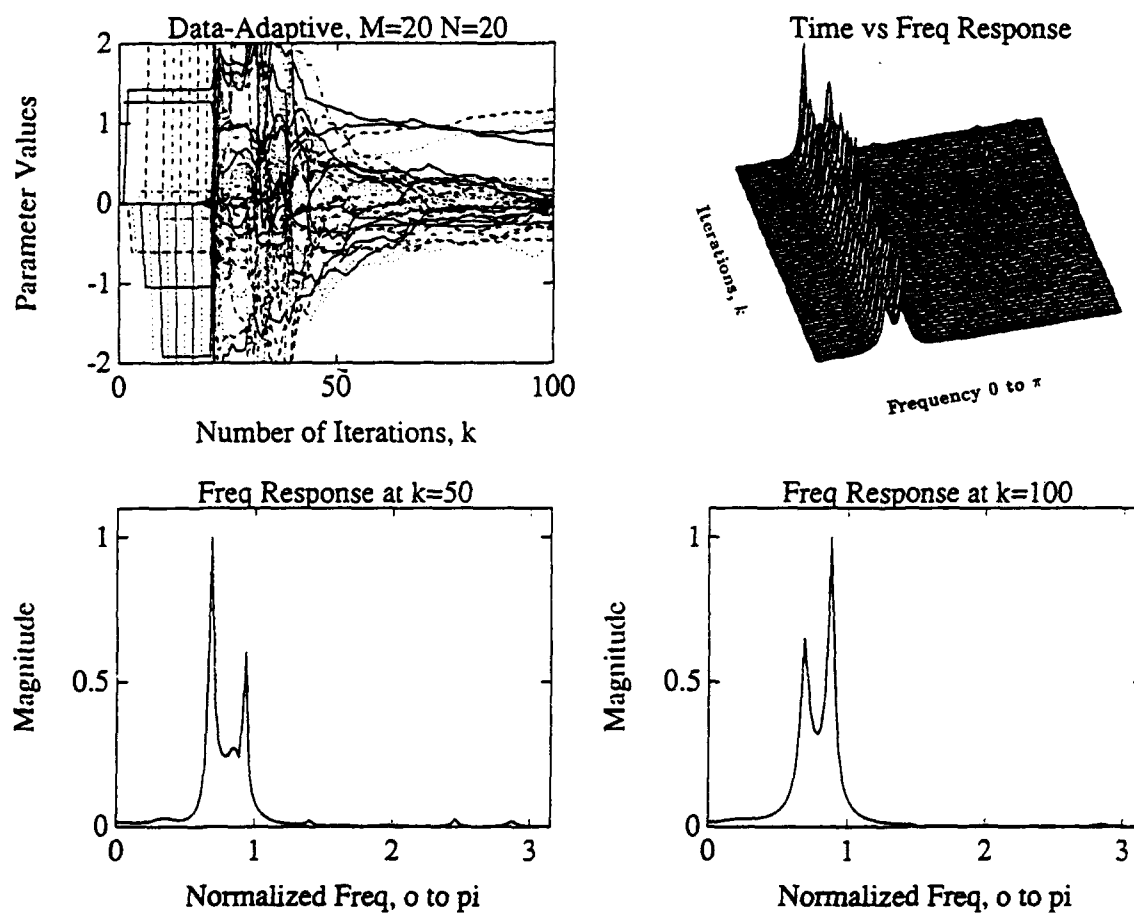


Figure C.8: Over-modeled example with 10 dB of additive noise for the data-adaptive Toeplitz approximation algorithm.

The 4th-order true system was modeled with a 20th-order model. The additive white noise tends to slow down the algorithm, but only slightly distorts the frequency response it is able to achieve. There was 10 dB of white noise added to the output.

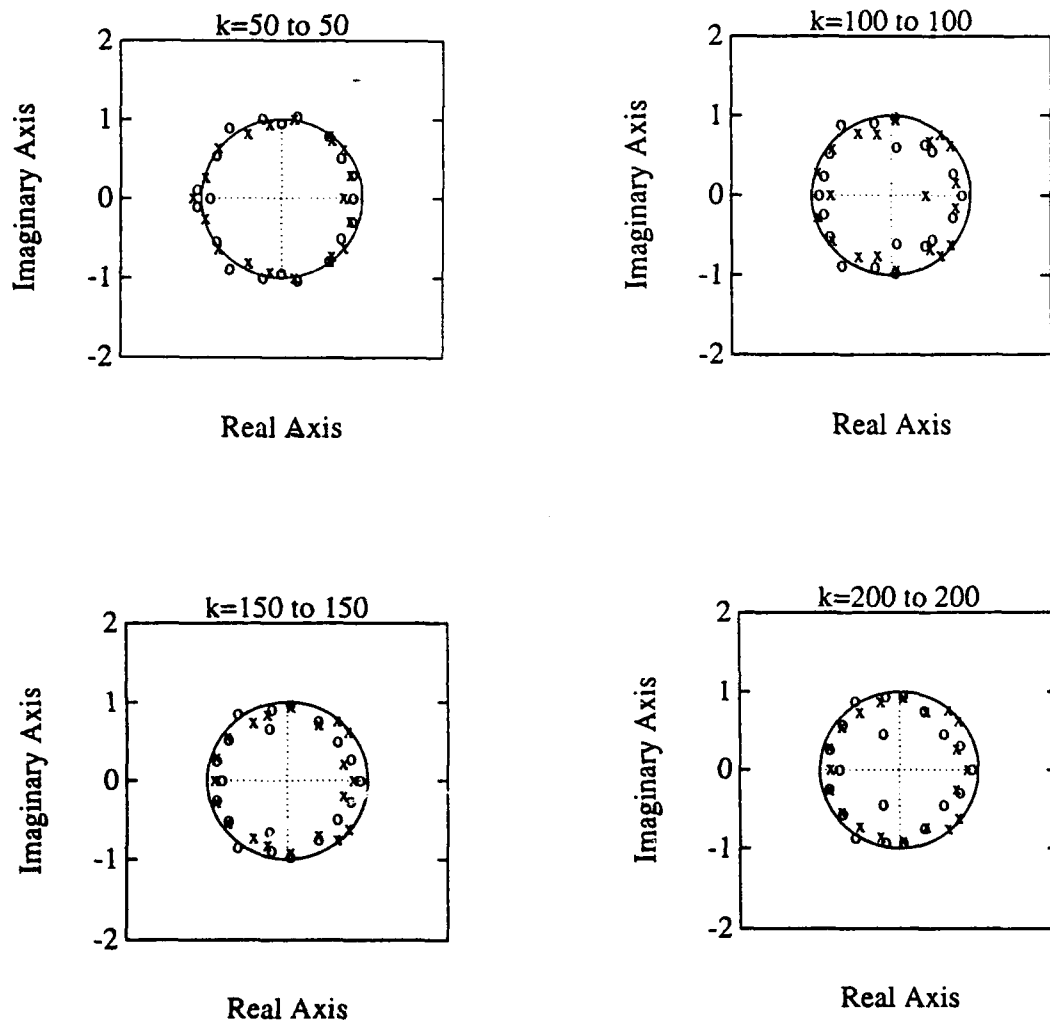


Figure C.9: Pole-zero plots for the over-modeled data-adaptive Toeplitz approximation algorithm with 10dB additive noise.

Observe that as the algorithm converges, the poles (x's) are all inside the unit circle. After about 100 iterations their location corresponds to the frequency response shown in Figure B.16. Again, we see that the pole-zero pairs different from those shown in Figure B.11 are almost overlapping and tend to cancel each other, which explains why both of the dominant frequency components in Figure B.10 are present.

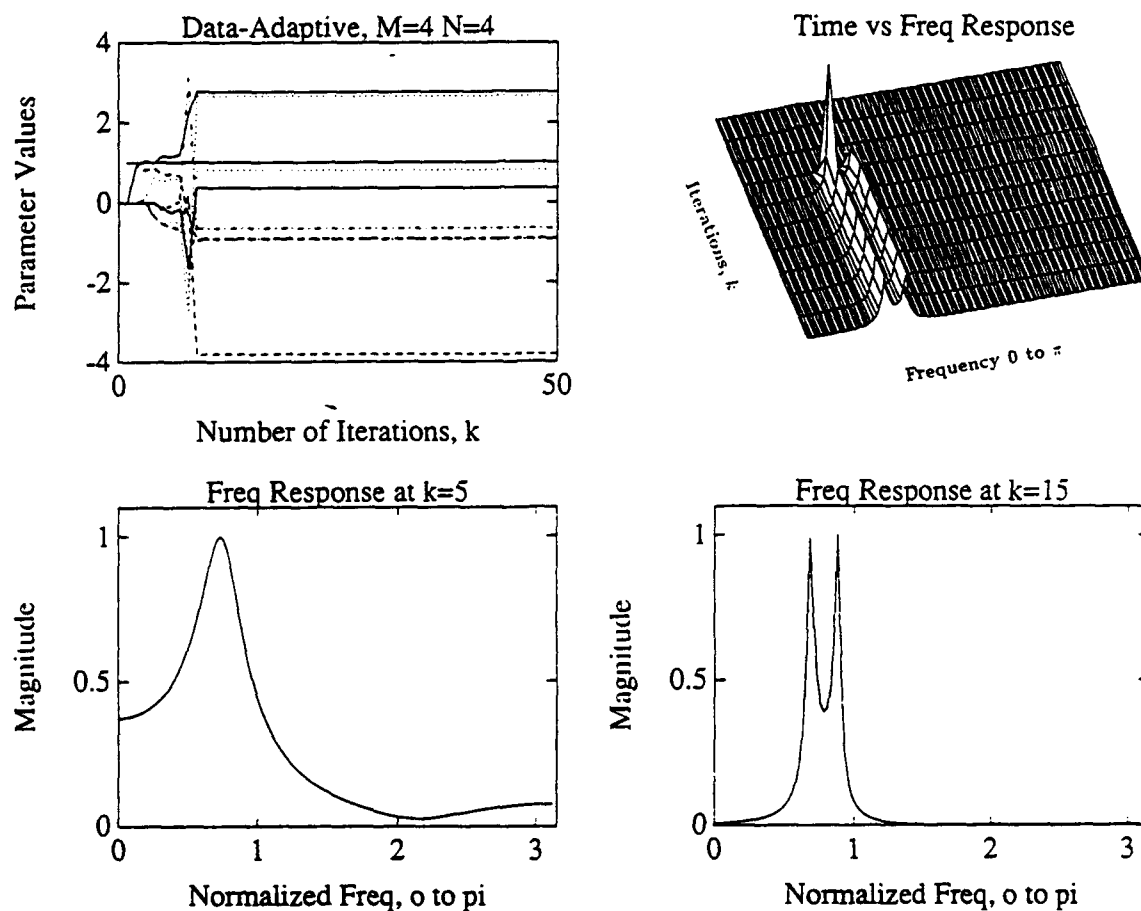


Figure C.10: Parameter tracks and frequency response of the IIR RLS algorithm.

Observe that the algorithm converges to the true solution and frequency response in ten iterations. In this example no noise was added to the output, and the performance above is for an exact modeling of the 4th-order true system. Comparing this with the data-adaptive Toeplitz approximation algorithm results shown in Figure B.10 reveals that they behave differently.

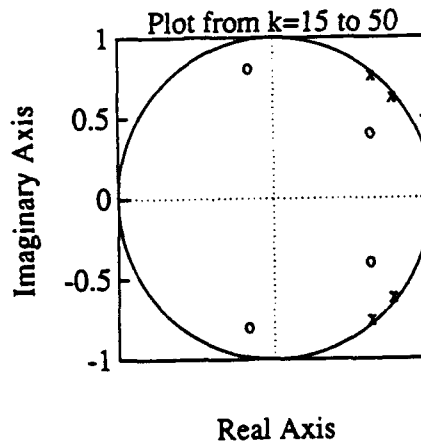
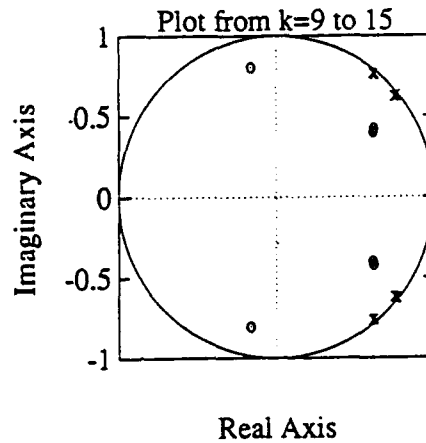


Figure C.11: Pole-zero plots for the IIR RLS algorithm.

Observe that as the algorithm converges to the true solution, the poles (x's) move inside the unit circle. After about 15 iterations they stop moving, and their location corresponds to the frequency response shown in Figure B.18. Finally, we see that there is almost no movement as the algorithm iterates from $k = 15$ to 50 iterations, which indicates that the algorithm has stabilized.

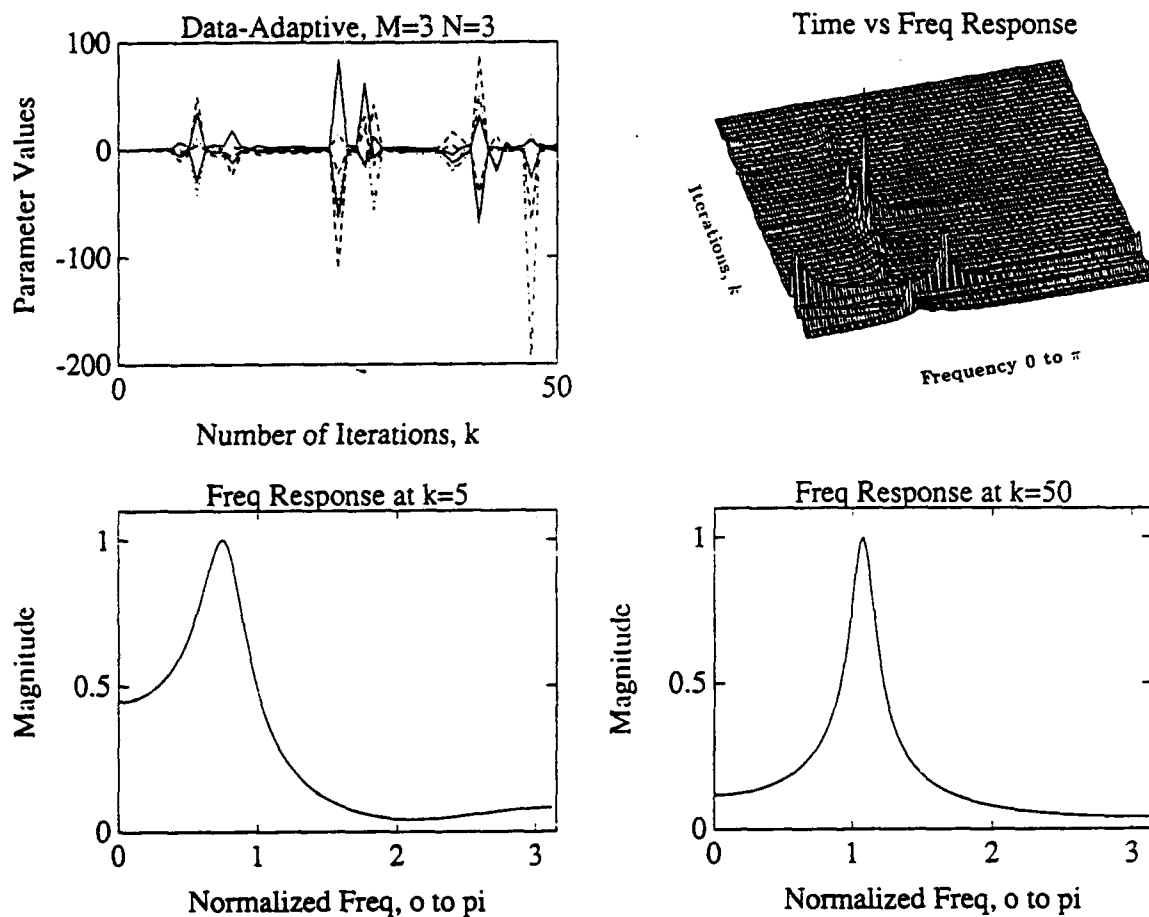


Figure C.12: Under-modeled example of the IIR RLS algorithm. The 4th-order true system was modeled with an 3rd-order model. The algorithm tries to preserve the frequency components of the true system, however it becomes unstable after about 50 iterations. No noise was added to the output.

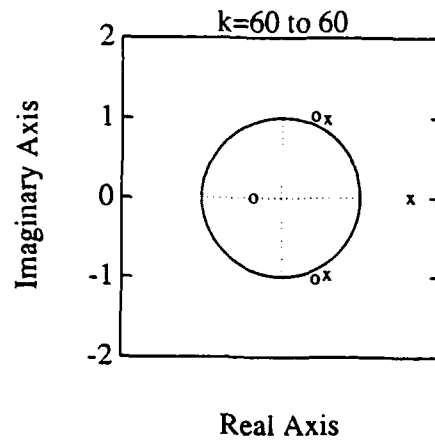
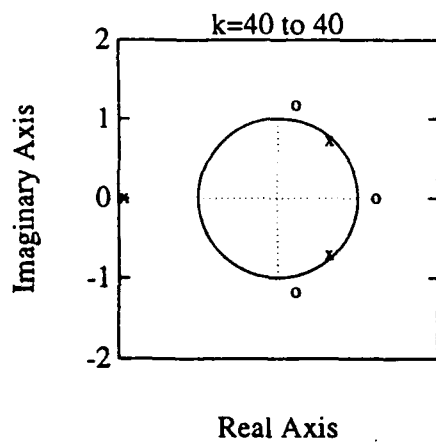
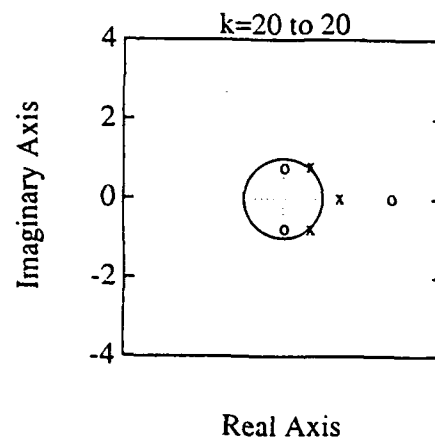
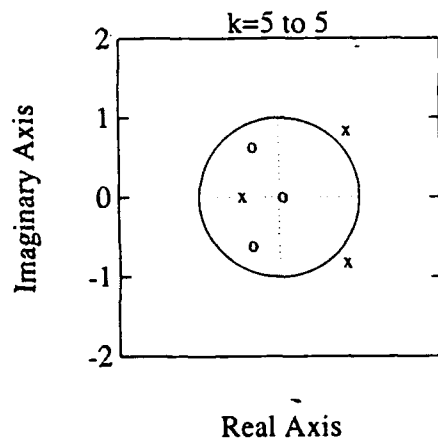


Figure C.13: Pole-zero plots for the under-modeled IIR RLS algorithm. Observe that as the algorithm converges, the poles (x's) are still outside the unit circle. After about 50 iterations their location reflects the fact that the algorithm is unstable, and it fails to converge.

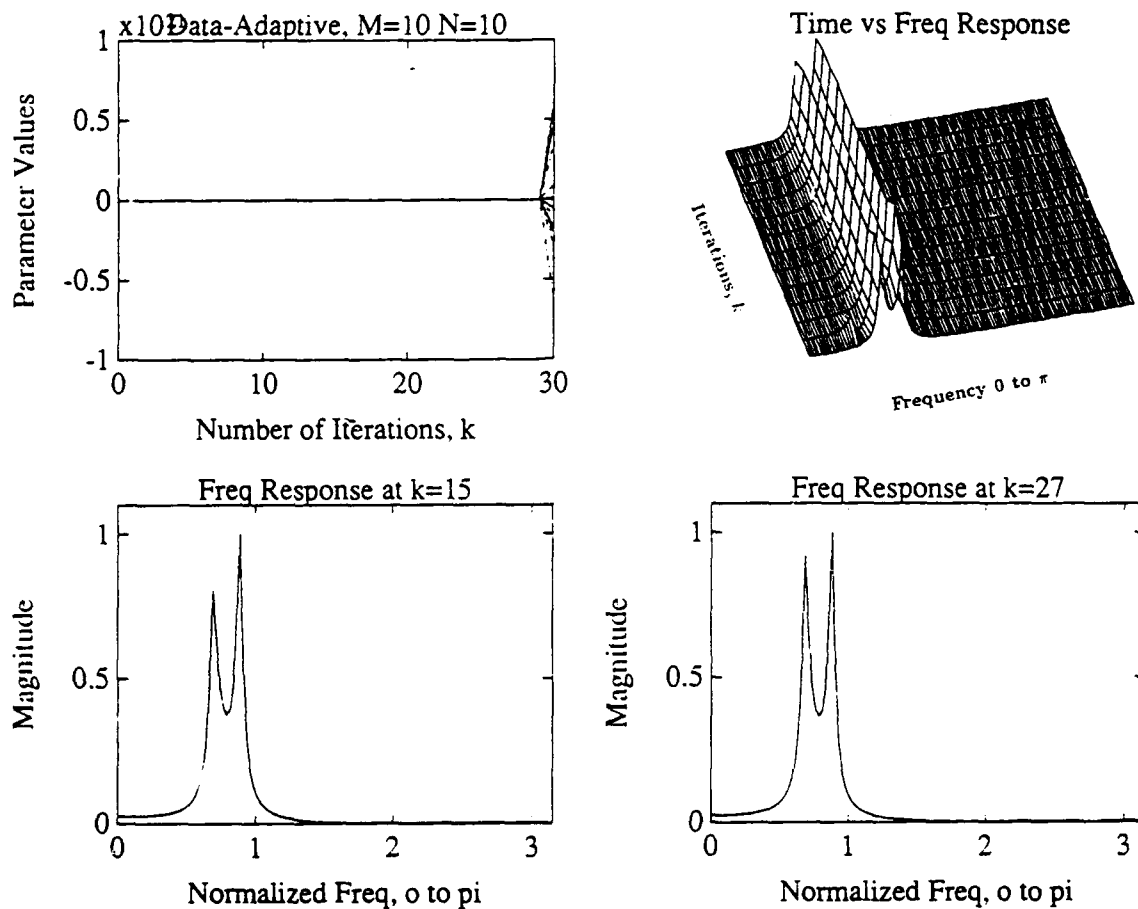


Figure C.14: Over-modeled example of the IIR RLS algorithm. The 4th-order true system was modeled with a 10th-order model. The algorithm also fails to converge, as can be seen after about 29 iterations in the parameter track plot. It is worth noting that for a brief number of iterations, from about $k = 10$ to 27, the algorithm does produce the true system frequency response. Unfortunately, this is not a stable result of the algorithm. No noise was added to the output.

APPENDIX D: MATLAB CODE FOR IIR ADAPTIVE ALGORITHM

```
% This is the IIR ADAPTIVE algorithm which uses the Toeplitz Approximation  
% splitting  $R=T+S$ .
```

```
N = input('How many poles in the system model would you like, N = ? ');  
M = input('How many zeros in the system model would you like, M = ? ');  
k = input('Enter the number of iterations (i.e. k = 100). ');  
noise = input('How much noise would you like (i.e., noise = .1) ? ');
```

```
maxMN = max(M+1,N);
```

```
MN = M + N;
```

```
s = .0000001; % A small amount used to start the algorithm below.
```

```
L = 1;
```

```
P = 1; % Start with one data point.
```

```
% Now generate the "true" system data.
```

```
rand('normal');
```

```
rand('seed',0);
```

```
a = [1,-2.76,3.809,-2.654,.924]; % These are the true system
```

```

b = [1,-.9,.81,-.65,.36];           % coefficients.

x = [zeros(M+1,1);rand(P + k,1)]; % The input data, white noise.
y = filter(b,a,x);                   % The output data.
x = flipud(x);                       % Reorder the input for convenience.
rand('seed',5);                      % A separate noise generator.
y = y + sqrt(noise) * [zeros(M+1,1);rand(P+k,1)]; % Add noise to output.
y = flipud(y);
y = [y(1:P+k,1);zeros(maxMN+1,1)]; % The additional zeros are used in the
x = [x(1:P+k,1);zeros(maxMN+1,1)]; % computation of Z below, read it.

% Now I will construct data matrix, Z

Z = zeros(P,MN + 1);

for i = 1:P
Z(i,:) = [y(i+k+2:N+k+1+i)',x(i+k+1:i+k+M+1)'];
% The counting index
% ensures that the algorithm starts with zero data points.
end

% Now generate the data correlation matrix, R,
% and partition it into 4 parts given by M and N.

R = Z' * Z;
Ryy = R(1:N,1:N);

```

```

Ryx = R(1:N,N+1:M+N+1);
Rxy = Ryx';
Rxx = R(N+1:M+N+1,N+1:M+N+1);

% Now generate the Toeplitz approximation to Ryy and Rxx.

tyy = zeros(N,1);
txx = zeros(M+1,1);

for i = 1:N
    tyy(i) = mean(diag(Ryy,i-1));
end

for i = 1:M+1
    txx(i) = mean(diag(Rxx,i-1));
end

Tyy = toeplitz(tyy) + s * eye(N);    % s is added so that
Txx = toeplitz(txx) + s * eye(M+1);  % these may be inverted.

Tiy = inv(Tyy);
Tix = inv(Txx);

% Now generate the cross-correlation vector, r,
% and partition it into two parts given by M and N.

```

```

r = Z' * y(2+k:P+k+1);
ry = r(1:N);
rx = r(N+1:M+N+1);

% Now compute the initial estimates of the parameter vectors.

b = Tiy * ry;
a = Tix * rx;

Qk = zeros(k,M+N+1); % Holds the results of each iteration.

for i = 1:k;          % This is the algorithm!!!

Xn = y(k+3-i : k+N+2-i); % get new output data
Xm = x(k+2-i : k+2+M-i); % get new input data
Yn = y(k+2-i);          % get next output data point

% Use the matrix inverse lemma to compute the next
% values for Tiy and Tix, note that L=1.

Tiy = (Tiy - ( Tiy * Xn * Xn' * Tiy) / (L+Xn'*Tiy*Xn))/L;
Tix = (Tix - ( Tix * Xm * Xm' * Tix) / (L+Xm'*Tix*Xm))/L;

% Update all of the rest of the time-varying quantities.

Ryy = Ryy + Xn * Xn';

```

```
Ryx = Ryx + Xn * Xm';
```

```
Rxy = Rxy + Xm * Xn';
```

```
Rxx = Rxx + Xm * Xm';
```

```
ry = ry + Xn * Yn;
```

```
rx = rx + Xm * Yn;
```

```
% Compute the next value of the parameter vectors.
```

```
b = b + Tiy * (ry - Ryx * a - Ryy * b);
```

```
a = a + Tix * (rx - Rxy * b - Rxx * a);
```

```
Qk(i,:) = [b',a'];
```

```
end;
```

REFERENCES

1. M. Tummala, "Efficient Iterative Methods for FIR Least Squares Identification," *IEEE Trans. on Acoustics, Speech, Signal Processing*, Vol. ASSP-38, No. 5, pp. 887-890, May, 1990.
2. M. Tummala, "Iterative Method for ARMA least Squares Identification," *IEEE Trans. Acoustics, Speech, Signal Processing* (submitted in the revised form).
3. Martin Mandelburg, *Systematic Experimental Determination of Discrete-Time models for Nonlinear Systems*, Ph.D. Dissertation, Naval Postgraduate School, Monterey, California, June 1982.
4. Simon Haykin, *Adaptive Filter Theory*, p. 313, Prentice Hall, 1986.
5. John J. Shynk, "Adaptive IIR Filtering," *IEEE Trans. on Acoustics, Speech, Signal Processing*, Vol. ASSP-6, No. 2, pp. 4-21, April, 1989.
6. Charles W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Prentice Hall, (to be published in 1991).
7. Gilbert Strang, *Linear Algebra and Its Applications*, pp. 297-303, Academic Press, New York, 1980.
8. Raymond H. Chan, "The Spectrum of a Family of Circulant Preconditioned Toeplitz Systems," *SIAM J. Numer. Anal.*, Vol. 26, No. 2, pp 503-506, April 1989.
9. David M. Young, *Iterative Solution of Large Linear Systems*, New York, NY, Academic Press, 1971.
10. Simon Haykin, *Adaptive Filter Theory*, p. 385, Prentice Hall, 1986.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code EQ Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor Murali Tummala, Code EC/Tu Naval Postgraduate School Monterey, California 93943-5000	5
5.	Professor Charles W. Therrien, Code EC/Th Naval Postgraduate School Monterey, California 93943-5000	1
6.	Commandant, USASC U.S. Army Signal School and Fort Gordon Fort Gordon, Georgia 30905	2
7.	Dr. R. Madan (Code 1114SE) Office of Naval Research 800 North Quincy Street Arlington, Virginia 22217-5000	1

8. U.S. Department of Transportation Library
Code M-493.3, Room 2200
400 7th Street South-West
Washington, D.C. 20590

1